Adaptation in Deep Learning Models: Algorithms and Applications

Christian Simon

A thesis submitted for the degree of
Doctor of Philosophy
The Australian National University

February 2022

I declare that this thesis has been composed by myself based on my prior research outcomes including previously published and pre-printed papers and the works have not been submitted, in whole or in part, for any other degrees at any other university. The works contained herein are my own in collaboration with co-authors and collaborators except explicitly stated otherwise.

Christian Simon
2 February 2022

To My Family
To My Forever Father

Acknowledgments

First and foremost, I would like to express my gratitude to the God Almighty as the works of the Lord are great; they are genuine delights for me to ponder. Next, during my study, I am grateful for my supervisors, Mehrtash Harandi, Piotr Koniusz, and Richard Hartley for their support and guidance. From day one, Mehrtash could point me in the right direction and challenge me to think of exciting research problems. Having him as my supervisor was one of the best decisions in my study as he always helped me to find solutions whenever I was stuck. I could remember many occasions that I was out of the track but he reminded me about good science. To Piotr, I thank that we had fascinating discussions, and he always encouraged me to push my boundaries and achieve maximum. In our collaboration, I could improve my research skills substantially. My gratitude is also equally directed towards my panel chair, Richard Hartley who kindly supports my research and administrative tasks related to my candidature at the Australian National University (ANU).

In retrospect over the past years of my Ph.D. study, I have met amazing people that are quite supportive and cooperative. I would like to acknowledge my collaborators at ANU and Data61, Richard Nock, Lars Petersson Pengfei Fang, Jie Hong, Ali Cheraghian, and Yan Han. We had wonderful discussions and fruitful collaborations. I also thank my other colleagues, Kartik Gupta, Amir Rahimi, Zhiwei Xu, Soumava Kumar Roy, and Samitha Herath for a cherished time spent together in office.

I am thankful to my family for their support and encouragement through my study. My mother has always given me endless encouragement to achieve the best version of myself. My brothers and sister have also fully-supported me in pursuing higher education. I also express my sincere thanks to my special one, Cindy Tang who has always been supportive in all time of my academic research.

Finally, I also thank my friends Ravi Arief, Laura Jenifer, Catherine Christianty, Suryati Gollinger, Hugo Sahetapy, Bianca Bongo, Praise Ichtus, Elisa Stephanie, Renata Purba, Mirelle Harsono, Calvin, Wendy Gunawan, Marcellina Go, Swen Kuh, Phyu Ang, Kristina Gallego, Anirvan Divakera, Wanissa Suanin, and Sungyeon Hong for wonderful time in Canberra.

Abstract

Artificial intelligence has been successful to match or even surpass human abilities *e.g.*, recognizing images, playing games, and understanding languages. This rapid progress is driven by machine learning models that can make use of a huge collection of data. Nevertheless, the nature of data in the real world scenario is very complex and changing depending on environments. Some aspects that affect the changes are the period of time, the relationship between objects, or the spatial locations. As a result, the data is quite dynamic and sparse, and this causes a limitation in building powerful models. At the current state, powerful machine learning models learn from data under a stationary environment while humans are capable of learning in dynamic, changing, and sequential conditions. In pursuing the idea of open-ended learning for machine intelligence, we contribute to provide algorithms and analyses for generally capable models via adaptation.

In this thesis, the model adaptation problem is defined as the impediment of intelligent machines to learn to modify their behaviors for new purposes or new uses. The ultimate goal is to develop machine intelligence that has the ability to adapt itself by not only following at our behest but also understanding the environment. Our works populate in the area of deep neural networks and transfer learning. Throughout our works, developing adaptive models is divided into four major problems: (1) few-shot learning, (2) fast model adaptation, (3) continual learning, and (4) architecture search. In few-shot learning, a model is expected to change its behavior when facing a new context or an unseen task with limited data. Another important problem within few-shot learning is to adapt quickly from a few data. In the problem of continual learning, the model needs to adapt sequentially depending on the given task. In architecture search, we look for a high-performing configuration for connecting among nodes in a model.

To approach the problem of few-shot learning, we opt to use the strategy in transfer learning with a pretrained Convolutional Neural Network (CNN) for novel tasks with limited-data annotations. Inspired by the success of subspace methods for visual recognition, we develop a classifier using subspaces to improve the generalization capability to novel concepts. We also investigate few-shot learning in multi-label classification, and propose a multi-label propagation technique by constructing a graph from the representations of support samples.

In pursuing fast model adaptation, we use the idea of preconditioners in optimization. Specifically, the problem revolves in *meta-learning*, where the agent needs to learn a family of tasks and adapt quickly to a new task. Our algorithm uses a non-linear function to generate the preconditioner for modulating the gradient when updating the model. Our experiments show that the model converges more quickly than other types of preconditioners in the same problem.

In the problem of continual learning, the model needs to sequentially learn and adapt the network parameters for new tasks without forgetting the previously learned tasks. To this end, we investigate the knowledge distillation approach, where the old model guides the current model to find the balance between the current task and the prior tasks. Our approach models

the smoothness between two tasks using the geodesic flow, and the objective is to maximize similarity of the projected responses along the geodesic flow.

In neural architecture search, the optimal architecture depends on the task objectives. We observe that searching for an optimal architecture is not trivial while the data annotations is noisy. The study investigates the impact of label noise in obtaining the best performance when optimizing a neural architecture, while also reducing the performance deterioration because of overfitting to noisy labels. We use the mutual information bottleneck to design a noise injection module that can alleviate the impact of learning under label noise.

In summary, our works in this thesis address some major problems in model adaptation *e.g.*, few-shot learning, meta-learning, continual learning, and neural architecture search. The solutions are expected to contribute to the arsenal of model adaptation algorithms and the analyses shed light on the essential aspects in adaptation strategies.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Perfection is like chasing the horizon.
Keep moving.

Neil Gaiman

1.1 Motivation

Suppose you have taken a cooking course. You will find that some cooking tools (*e.g.*, solid turner, whisk, and grater) are new to you. You might not need to learn cooking skills from the beginning as you have some experience in basic cooking *e.g.*, slicing, peeling, or stirring. For some new tools, you might guess how to use these alien tools because you have some experience using similar objects. Once you have learned to use the tools then cooking a dish following a recipe is at your fingertips. This learning experience is a kind of adaptation to acquire a new skill.

We, as human beings, find ourselves in a developmental phase by learning new concepts everyday. The term *learning* is very broad, and we refer to the modern definition of by Mitchell [Mitchell, 1997]: "A computer program is said to *learn* from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E". This thesis concentrates in skill learning[1]. In our life-span, we learn and accumulate skills even from a few examples. Consider children learn several different tasks *e.g.*, stand, walk, run, and jump but they never learn these activities from scratch. Central to this human's learning capability is *adaptation*, which is defined as the ability to acquire novel skills and alter behavior as a result of experience. Human's brain is trained with the capability to transfer gains to other skills within the same family of tasks. For instance, learning to run requires only some adaptation after learning to walk without fundamental changes on the movement.

In general, the works in this thesis contribute to develop machine learning models that take inspiration from human learning mechanisms. An ultimate goal for this research aims to build the foundation for life-long learning, *i.e.*, an intelligent machine can continue to learn throughout their lifespans. To engage in some complex and changing environments, an intelligent machine

[1] Skill learning is defined as an improvement, in perceptual, cognitive, or motor performance as a result of training that persists for a period of time [Green and Bavelier, 2009].

needs to adapt by leveraging previous knowledge and experience. Particularly, in the era of Deep Neural Networks (DNNs), a neural network stores some parameters to cover a broad range of tasks and the network's weights have to be adjusted depending on a specific task. Imagine a general-purpose robot that could help humans to finish their tasks *e.g.*, cooking, dish-washing, and cleaning would benefit from this line of research.

To achieve the human ability to learn quickly, adaptive, and generalize robustly, a model needs to have capabilities to learn in complex and changing environments. To this end, we breakdown the solution into four sub-topics: few-shot learning, meta-learning, continual learning, and Neural Architecture Search (NAS). Throughout this thesis, we consider five distinct problems:

1. We consider learning with limited annotations, where there are not a lot of samples available to train the model. Moreover, we also consider the few-shot learning problem viewed through the lens of multi-label classification in one instance.

2. We explore the adaptation capability for fast model adaptation, which is an important property in learning-to-learn.

3. In the scenario of lifelong learning, a model is sequentially updated when there is a stream of tasks, also known as continual learning.

4. In real world applications, a neural network structure might change depending on the task. In this problem, we investigate the neural architecture search algorithm in the presence of label noise. By and large, the contribution to make a robust searching algorithm is expected.

1.2 Neural Networks in a Changing Environment

One may believe that adapting a DNN to a new environment is not difficult. Below, we provide some examples to show the difficulty of adaptation but let us first fix some terminologies. In the context of this thesis, an environment can be described as a set of stimuli or examples. The adaptation based on stimuli is also known in the literature as learning. The aim is to find the function of a DNN that describes the input-output relation. As a result, a neural network can capture the representation of the environment. In classical implementation of neural networks, the assumption is that the data is stationary meaning that train and test datasets are in the same environment. For instance, we want to build a model that can classify cats and dogs, the images for training and testing are collected in the same time frames. In this case a DNN must "optimally" suit the outcomes of a fixed unchanging environment.

Despite the success of deep neural networks in image recognition [He et al., 2016; Dosovitskiy et al., 2021; Krizhevsky et al., 2017], natural language processing [Sutskever et al., 2014; Devlin et al., 2019], and robotics and gaming [Mnih et al., 2015; Zhu et al., 2020], it is still challenging to create brain-inspired models that can continuously learn, reason, and quickly understand new concepts from a few observations. The limitation of such existing models is a lack of capabilities to be dynamic. The performance trained deep neural networks (without data augmentation) easily drops when there is simple noise added to the input image [Kurakin

et al., 2017]. In real-world scenarios, models must be robust to a changing environment *e.g.*, lighting change for image recognition, preference change in rewards for robotics, and change of contextual meaning for natural language processing. In particular, the works in this thesis tackle the problem of adaptation in a changing environment, meaning the model needs to alter their behaviors to novel stimuli. We consider that the learning process has concept drifts where there is a transition from one environment to another environment. The assumption made in the context of learning with a changing environment is that the past experience can contribute to acquire new skills while adjusting a model to future observations and outcomes in a new environment. The learning procedures for a changing environment are various in the real-world scenarios *e.g.*, training with unlabeled data with domain shifts, a few examples from unseen classes, or less access to past examples.

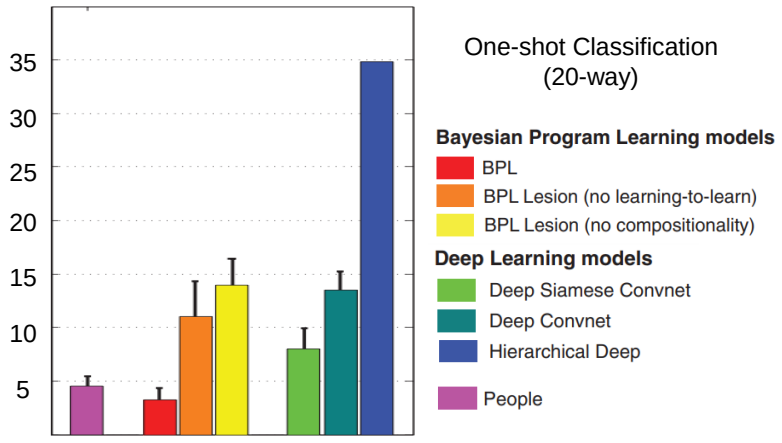


Figure 1.1: A comparison of deep learning models, Bayesian Program Learning (BPL), and humans in recognizing new characters. The error rates of human and BPL are below 5% to classify new characters from a few observations but the deep learning models perform worse. (Note: This image is taken from [Lake et al., 2015a])

Suppose our goal is to build a neural network that can learn from a few images that are never seen previously. An example is the experiments conducted in [Lake et al., 2015b] to recognize characters from various languages. In this setting, the model learns with a limited amount of examples *e.g.*, one shot per character. The purpose is to acquire new skills to recognize unseen characters. Human ability for the aforementioned task is compared against deep neural networks and Bayesian Program Learning (BPL). Figure 1.1 shows that deep learning models cannot surpass human's performance when the data is novel and limited. In contrast to the results on ImageNet [Krizhevsky et al., 2012] using the ResNet structure [He et al., 2016], the performance of standard DNNs is comparable to human-level performance. This is because standard DNNs are data hungry, and the number of samples is not enough to achieve desired performance for novel concepts.

1.3 Adaptation in Neural Networks

For new tasks and environments, our biological brains do not have to learn from scratch. This observation brings the idea of transferring the knowledge or skills from past experience for neural networks. The transfer of knowledge is the key for reusing knowledge in previous learning tasks. In literature, the framework to benefit from previously-acquired skills or knowledge is known as transfer learning [Pan and Yang, 2009]. In neural network adaptation, the scenarios range from a short time frame adaptation *e.g.*, network updates for recognizing a new object to a life-long adaptation *e.g.*, continually update the networks when encountering a stream of learning new objects across the lifespan of intelligent agents.

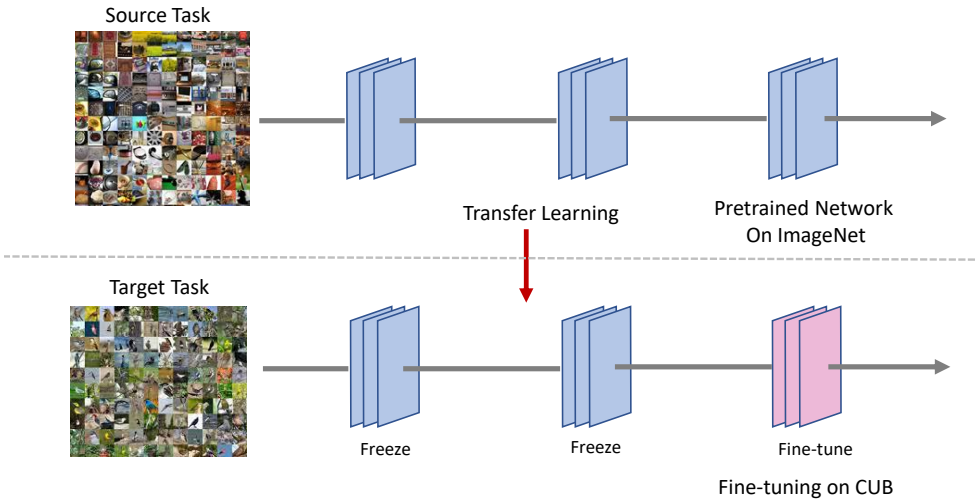


Figure 1.2: An illustration of fine-tuning a DNN. The pretrained DNN on ImageNet [Krizhevsky et al., 2012] is transferred to the target task for recognizing birds [Welinder et al., 2010] by fine-tuning the last layer of the pretrained DNN.

Adaptation strategies for deep neural networks. To this point, the description of how to perform transfer learning is an abstract way. Let us deep dive into the DNN adaptation in practice and how the structures and parameters of a DNN are modified. The adaptation strategies are divided based on how the DNNs are updated and manipulated in the training phase.

—**Fine-tuning a pretrained DNN.** A straightforward approach for model adaptation is by fine-tuning the parameters after learning from the first task or a set of data. The intuition is that transfer of knowledge improves the generalisation of DNNs to novel tasks. The key difference compared to standard learning is that fine-tuning requires less iterations to converge for similar tasks because the parameters are well-initialized. The empirical evidence in [Thrun, 1995] demonstrates that past knowledge acquisition makes learning new tasks easier, particularly when training data is limited or scarce. In computer vision, natural language processing, this approach is widely used to apply a pretrained deep neural network (*e.g.*, on foundation models trained on very large datasets [Bommasani et al., 2021; Yuan et al., 2021]) for the downstream tasks *e.g.*, object detection and video classification. For instance, the pretrained network on

ImageNet [Krizhevsky et al., 2012] is fine-tuned by learning new samples from the Caltech-UCSV Birds (CUB) dataset [Welinder et al., 2010] as illustrated in Figure 1.2. Even though the fine-tuning process usually considers labelled data, but there are also a case with unlabelled data as in unsupervised domain adaptation Ganin and Lempitsky [2015]; Li et al. [2020]. The problem is defined as learning from a new target domain with overlapping

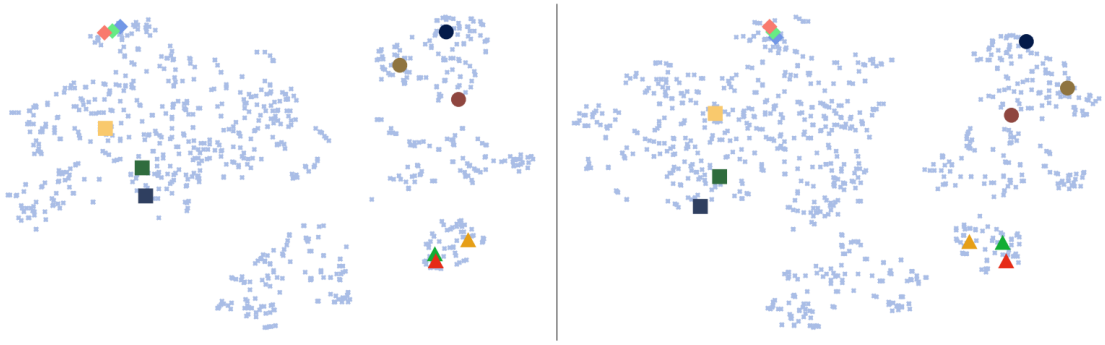


Figure 1.3: A visualization using t-SNE [van der Maaten and Hinton, 2008] on the activations of each category before the fully connected layer of a pre-trained ResNet-50 [He et al., 2016] (left) and the fully connected layer parameters (right). Category is represented as a point and the highlighted points with the same color and shape comes from the same category. (Note: The image is taken from [Qiao et al., 2018])

—**Modifying a classifier.** In a standard DNN for supervised learning, the last layer is usually equipped with Fully-Connected (FC) layers. However, having FC layers becomes problematic if we cannot update them during adaptation for a new task. Classifiers trained on a specific task are not transferable to a new task, but representations of CNNs are transferable to a new task [Oquab et al., 2014]. In [Qiao et al., 2018], the motivation for dynamic classifiers using the learned representations is presented to replace standard classifiers using FC layers. Figure 1.3 shows that the features extracted by a pretrained Convolutional Neural Network (CNN) have a relationship with the parameters from the FC layer. This visualization shows that the neighbor embeddings of averaged features of each class and the FC layer parameters share similarity in local and global structures. Using this strategy, it is not necessary to update DNNs but the classifier is made dynamic depending on the learned representations. To use the existing FC layer trained on initial/base categories, we can add the averaged features from samples of novel categories. In our work, we make use of second-order classifiers that are constructed from a few samples and present a class representation in a low-dimensional subspace.

—**Updating network parameters for multiple tasks.** Replacing the classifier is very common for image classification tasks [Qiao et al., 2018; Simon et al., 2020a; Gidaris and Komodakis, 2018a]. However, there are some other tasks in reinforcement learning and regression that need to update the parameters in neural networks. The adaptation through this technique updates the parameters using any standard gradient descent methods *e.g.*, Stochastic Gradient Descent (SGD) or Adam [Kingma and Ba, 2015]. When there are multiple tasks with different behaviors, we would like to have a high performing model for each task. Let us take an example in learning multiple tasks by finding the initial neural network parameters such that learning a new task

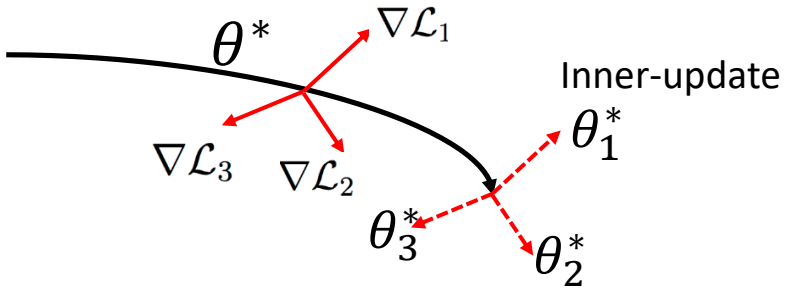


Figure 1.4: An illustration of learning to find optimal parameters based on a specific task (also known as meta-learning). Each task has its own optimally adapted model parameters $\theta_1^*, \theta_2^*, \theta_3^*$ updated from the initial model parameter θ^* using gradients $\nabla\mathcal{L}_1, \nabla\mathcal{L}_2, \nabla\mathcal{L}_3$.

.

requires less updates. Model Agnostic Meta-Learning (MAML) [Finn et al., 2017] adopts this approach to achieve good generalization performance for multiple tasks including regression, robot policy finding, and image classification. Figure 1.4 illustrates how the algorithm works to achieve optimal parameters for each task using gradient descent.

—**Learning with multiple models.** In reality, we also learn not only from our own observations

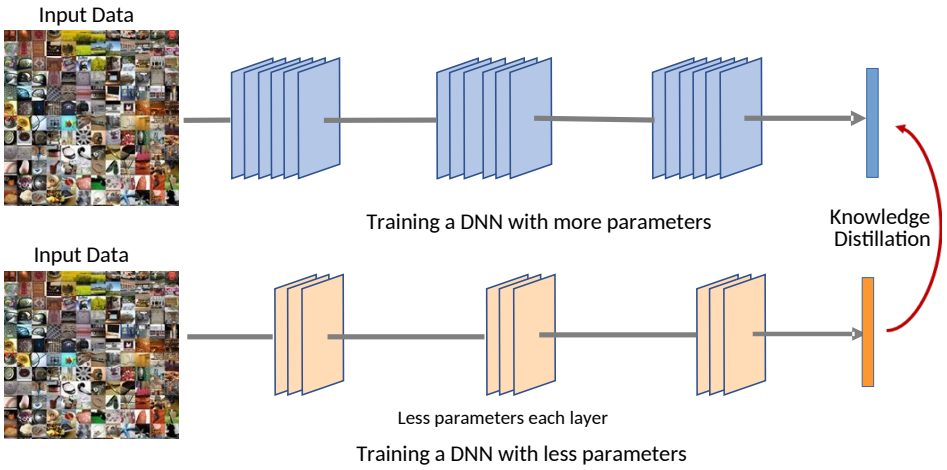
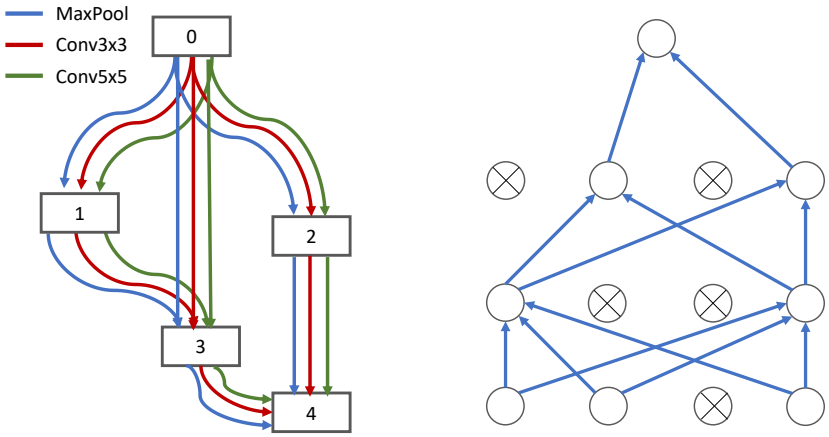


Figure 1.5: An illustration of learning with another DNN with the teacher-student model where the network with less parameters (student) is trained using the outputs of the network with more parameters (teacher).

.

but also based on knowledge provided by the other people, experts, or teachers. Hereafter, a person who provides such knowledge is called a teacher. The knowledge that we acquire, in this case, might be distilled following the behavior of the teacher. The teacher imparts some knowledge while we also learn new tasks. The adaptation strategy following this idea uses two or more models and matches the outputs of the teacher network and the student network

as shown in Figure 1.5. This knowledge distillation approach is initially used to produce a manageable size of DNNs in [Hinton et al., 2015]. This solution can be viewed as adaptation with additional constraints for efficient neural networks, and it serves a different paradigm compared to *ensemble* methods in machine learning. In practice, this technique contributes to a broad range of problems in machine learning *e.g.*, neural networks compression and compact networks [Chen et al., 2017; Sau and Balasubramanian, 2016; Liu et al., 2019c], semi-supervised learning [Tarvainen and Valpola, 2017], domain adaptation [Zhao et al., 2020], and privacy aware training [Papernot et al., 2017]. The training strategy using a teacher model for knowledge distillation also improves the performance of self-supervised learning with data augmentation in [Caron et al., 2021; He et al., 2019].



(a) NAS (b) Targeted Dropout

Figure 1.6: An adaptation strategy by changing the network structure. (a) An illustration of NAS where the operations (*e.g.*, MaxPool, Conv3x3, and Conv5x5) are picked by training on a specific dataset to find a high-performing network structure. (b) An illustration of targeted dropout where the connections between nodes are selected or removed to create a subnetwork.

—**Altering the structure of DNNs.** The structure of a DNN can be altered depending on the environment. The idea is that the network should be adaptive in responding the changes in the environment. Some parameters are triggered more significantly for one task than another task. There are some variants of network structure manipulation that can be employed in practice. A common strategy to find an optimal strategy automatically is Neural Architecture Search (NAS) [Liu et al., 2019b; Zoph and Le, 2017; Zoph et al., 2018; Pham et al., 2018] in which the connections between nodes of a neural network are selected as shown in Figure 1.6. For multiple tasks, the strategy is to switch on and off the connections between nodes of two consecutive layers and update the neural network using gradient descent as proposed in [Elsken et al., 2020; Rajasegaran et al., 2019], and the most common technique to apply this method for a specific task is well-known as *dropout* [Srivastava et al., 2014]. The structure can also be modified using addition and removal of some parameters during training as described in [Rusu et al., 2016; Wu et al., 2020; Li et al., 2019c].

1.4 Contributions

Considering the aforementioned strategies, we introduce algorithms that adopts adaptation in DNNs and provide relevant analysis to each algorithm. We identify the problems in each adaptation strategy and develop novel methods to address them. In particular, we aim to provide algorithms and applications for few-shot learning, meta-learning, continual learning, and NAS under label noise. We also investigate the intriguing properties of our proposed approaches and apply them in various conditions. Thorough analysis and comparison to the-state-of-the-art methods are provided to understand the proposed approaches in details. For all works in this thesis, the contributions on adaptation strategies of DNNs are listed below:

- In [Harandi et al., 2014; Basri and Jacobs, 2003], subspaces are used to address visual classification tasks that involve image sets. Inspired by this strategy, we make use of subspace methods for image recognition where high-dimensional visual inputs can be represented in low-dimensional spaces. For the problem of few-shot learning, we propose a subspace method where a classifier is dynamically constructed based on a subspace representing each class in an episode.

- As considered an effective approach for visual tasks and machine learning, we further extend our approach using the notion of Grassmann manifolds [Harandi et al., 2014, 2016, 2015] where we consider the subspace-to-subspace similarity between two sets. In our approach, the subspace is constructed based on the bag of features from image patches. As shown empirically, constructing a subspace using image patches improves discriminative power for few-shot learning.

- Moreover, we also analyze and study the robustness property of using subspaces as dynamic classifiers[2] for few-shot learning and also provide a computationally efficient approximation for subspace creation.

- In meta-learning, we propose a fast adaptation method using modulated gradients for a broad range of tasks such as image classification, image regression, and robotic navigation. We exploit the modulation to the gradients with some inspiration from optimization techniques using preconditioners.

- We propose a graph technique and a counting method for multi-label few-shot learning. The graph technique uses label propagation to weigh the features of each sample depending on the label and the neural label counting module to estimate the number of labels.

- In continual learning, we extend knowledge distillation in a feature space [Hou et al., 2019b] that calculates the distance of past and current representations using the Euclidean distance to alleviate forgetting past experience. We propose a novel knowledge distillation loss that takes into account the smoothness between two consecutive tasks. In detail, the knowledge distillation considers the geometry aspect of learning dynamics, and we exploit the geodesic connection (*i.e.*, the geodesic flow) between two tasks on the Grassmann manifolds.

[2]The 'dynamic' word means the classifier changes based on the contextual inputs.

- In searching a high-performing neural architecture, we study the searching process under label noise and exploit the noise injection method to design a resilient algorithm in searching an optimal architecture. We draw a connection of learning under label noise with the mutual information bottleneck principle. In short, the mutual information between latent variables and noisy labels should be suppressed to avoid overfitting. We also derive the mutual information bottleneck with a variational method as a lower bound, and this derivation yields a noise injection module used to reduce overfitting to noisy labels.

1.5 Thesis Outline

In this section, we provide a brief description about each chapter in this thesis. The background for related problems and relevant works are discussed in Chapter 2. The methods developed for few-shot learning and meta-learning are presented in Chapter 3, 4, and 5. The continual learning using geodesic distillation loss and the NAS under label noise are discussed in Chapter 6 and 7, respectively. Lastly, we conclude this thesis in Chapter 8.

—**Chapter 2 - Background:** This chapter introduces the problems in detail and discusses their related background. Relevant literature of few-shot learning, meta-learning, preconditioning in optimization, continual learning, neural architecture search, and learning under label noise is also discussed to describe the progress of each line of research. The description in this chapter also includes terminology, contributions of prior works, training and evaluation protocols, measurements, and adaptation in DNNs for each problem.

—**Chapter 3 - Subspace Methods for Few-Shot Learning:** This chapter presents a method for few-shot learning using subspaces. We provide a framework for few-shot learning by introducing dynamic classifiers that are constructed from few samples. In doing so, we make use of subspaces as the central block of a dynamic classifier. We empirically show that such modeling leads to robustness against perturbations (e.g., outliers) and noise, yielding competitive results with state-of-the-art algorithms. In addition to using subspace methods for point-to-subspace distance evaluations, few-shot learning is also investigated under a Grassmann perspective which provides a way to compute the distance between subspaces. This setting is used with a bag of features per image to boost the performance further. Finally, to cover several possible working regimes of our few-shot learning pipeline, we also explore kernelized subspace methods and approximate subspace computations, each showing different strengths.

—**Chapter 4 - Multi-Label Propagation for Multi-Label Few-Shot Learning:** In this chapter, we introduce the problem of multi-label few-shot learning (ML-FSL) which presents a more challenging task compared to the traditional multi-class few-shot learning. In order to address the ML-FSL problem, we propose a more general form of episodic learning and extend two celebrated FSL methods to work under the ML-FSL regime. We then propose a novel deep learning technique that makes use of label propagation to address the problem of ML-FSL. We also introduce a neural module to estimate the label count of a given sample by exploiting the relational inference. We will show empirically the benefit of the label count module, the label propagation algorithm, and the extensions of conventional FSL methods on three challenging

datasets, namely MS-COCO, iMaterialist, and Open MIC. Overall, our thorough experiments suggest that the proposed label-propagation algorithm in conjunction with the neural label count module (NLC) shall be considered as the method of choice.

—**Chapter 5 - On Modulating the Gradient for Meta-Learning:** In this chapter, we visit the problem of fast-adaptation in DNNs. Inspired by optimization techniques, we propose a novel meta-learning algorithm with gradient modulation to encourage fast-adaptation of neural networks in the absence of abundant data. Our method, termed ModGrad, is designed to circumvent the noisy nature of the gradients which is prevalent in low-data regimes. Furthermore and having the scalability concern in mind, we formulate ModGrad via low-rank approximations, which in turn enables us to employ ModGrad to adapt hefty neural networks. We thoroughly assess and contrast ModGrad against a large family of meta-learning techniques and observe that the proposed algorithm outperforms baselines comfortably while enjoying faster convergence.

—**Chapter 6 - On Learning the Geodesic Path for Incremental Learning:** This chapter presents a distillation method for incremental/continual learning to prevent *catastrophic forgetting*. Overcoming *catastrophic forgetting* is of significant importance to emulate the process of "incremental learning", where the model is capable of learning from sequential experience in an efficient and robust way. State-of-the-art techniques for incremental learning make use of knowledge distillation towards preventing catastrophic forgetting. Therein, one updates the network while ensuring that the network's responses to previously seen concepts remain stable throughout updates. This in practice is done by minimizing the dissimilarity between current and previous responses of the network one way or another. Our work introduces a novel method to the arsenal of distillation techniques. In contrast to the previous state of the art, we propose to firstly construct low-dimensional manifolds for previous and current responses and minimize the dissimilarity between the responses along the geodesic connecting the manifolds. This induces a more formidable knowledge distillation with smooth properties which preserves the past knowledge more efficiently as observed by our comprehensive empirical study.

—**Chapter 7 - Towards a Robust Differentiable Architecture Search Under Label Noise:** This chapter formally introduces the problem of searching a neural architecture under label noise. Previous studies focus on developing NAS algorithms for clean high quality data, a restrictive and somewhat unrealistic assumption. In this work, focusing on the differentiable NAS algorithms, we show that vanilla NAS algorithms suffer from a performance loss if class labels are noisy. To combat this issue, we propose to make use of the principle of information bottleneck as a regularizer. This leads us to develop a noise injecting operation that is included during the learning process, preventing the network from learning from noisy samples. Our empirical evaluations show that the noise injecting operation does not degrade the performance of the NAS algorithm if the data is indeed clean. In contrast, if the data is noisy, the architecture learned by our algorithm comfortably outperforms algorithms specifically equipped with sophisticated mechanisms to learn in the presence of label noise. In contrast to many algorithms designed to work in the presence of noisy labels, prior knowledge about the properties of the noise and its characteristics are not required for our algorithm.

—**Chapter 8 - Conclusion:** In the final chapter of this thesis, the overall contributions are summarized and future directions are discussed for this line of research.

1.6 Publications

The papers that are associated with the research works in this thesis are enlisted below. These publications have never been used to obtain a degree in another university or institution.

- **Christian Simon**, Piotr Koniusz, Richard Nock, and Mehrtash Harandi. Adaptive Subspaces for Few-Shot Learning. Published in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020.
- **Christian Simon**, Piotr Koniusz, Richard Nock, and Mehrtash Harandi. On Modulating the Gradient for Meta-Learning. Published in: European Conference on Computer Vision (ECCV), 2020.
- **Christian Simon**, Piotr Koniusz, and Mehrtash Harandi. On Learning the Geodesic Path for Incremental Learning. Published in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2021.
- **Christian Simon**, Piotr Koniusz, Richard Nock, and Mehrtash Harandi. Subspace Methods for Few-Shot Learning. Submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2021.
- **Christian Simon**, Piotr Koniusz, Lars Petersson, Yan Han, and Mehrtash Harandi. Towards a Robust Differentiable Architecture Search under Label Noise. IEEE Winter Conference on Applications of Computer Vision (WACV), 2022.
- **Christian Simon**, Piotr Koniusz, and Mehrtash Harandi. Multi-Label Propagation for Multi-Label Few-Shot Learning. IEEE Winter Conference on Applications of Computer Vision (WACV), 2022.

During my PhD, I also contributed to the following papers about few-shot learning and continual learning where I serve as co-author. These papers will not be discussed in this thesis.

- Ali Cheraghian, Shafin Rahman, Sameera Ramasinghe, Pengfei Fang, **Christian Simon**, Lars Petersson, and Mehrtash Harandi. Synthesized feature based few-shot class-incremental learning on a mixture of subspaces. International Conference on Computer Vision (ICCV), 2021.
- Jie Hong, Pengfei Fang, Weihao Li, Tong Zhang, **Christian Simon**, Mehrtash Harandi, and Lars Petersson. Reinforced attention for few-shot learning and beyond. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2021.

Chapter 2

Background

This chapter presents the background for the problems involved in adaptation of DNNs. Throughout this chapter, we bring clarity to the basic concepts of *task, meta-learning, episode, catastrophic forgetting*, and *noisy labels*. Firstly, we look at the problem of few-shot learning and its variant in multi-label learning. Then we focus on describing about fast adaptation in DNNs with the term called *meta-learning*. We also dive deep into incremental learning problems and the various methods to mitigate *catastrophic forgetting*. Lastly, we discuss the literature about neural architecture search and elaborate on the problem of learning under label noise.

2.1 Few-Shot Learning

2.1.1 Overview

Supervised learning using neural networks has achieved exceptional results in recognition problems, including visual recognition and language understanding. Predictive algorithms work well for labeled data using large computing resources. However, such methods depend heavily on large amounts of input data and annotations to create high-performing models. Initially, it seemed that labeling the data was not an issue but the growth of the data makes annotations problematic. As a result, newly incoming classes may only have a few samples to train or tune the model. In the presence of such conditions, standard DNNs without any adaptation based on the context is less generalizable to the unseen classes as shown in Table 2.1.

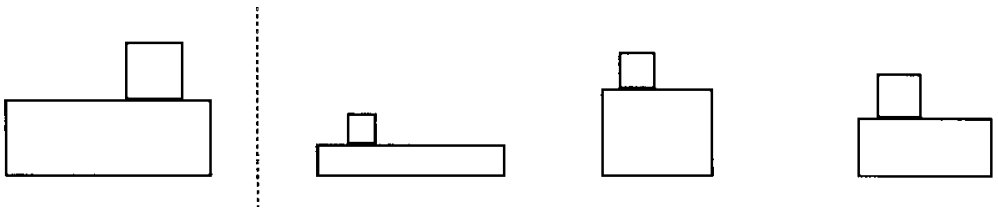


Figure 2.1: One-shot categorization. Given one example on the left side, humans can easily recognize novel examples of the category (right side) in a more general form. In this example, an observer may guess the right side belongs to the same category as the left one where they have a general form *i.e.*, a square on the top of a rectangle. (Note: This image is adapted from [Feldman, 1997])

Model	Matching Function	Fine Tune	5-way acc.	
			1-shot	5-shot
PXL	Cosine	No	23.0%	26.6%
Baseline Classifier	Cosine	No	36.6%	46.0%
Baseline Classifier	Cosine	Yes	36.2%	52.2%
Baseline Classifier	Softmax	Yes	38.4%	51.2%
Matching Nets [Vinyals et al., 2016]	Cosine (FCE)	Yes	46.6%	60.0%

Table 2.1: Accuracy on the miniImageNet dataset [Vinyals et al., 2016; Ravi and Larochelle, 2017] shows that learning from a few data needs a special training strategy to gain improvements against linear probing (only update classifiers). PXL calculates matching scores using the cosine similarity on raw pixels and Baseline Classifier uses the nearest neighbors matching. All these baselines are compared to Matching Network [Vinyals et al., 2016] which is a specially designed algorithm for few-shot learning with an *episodic* learning strategy. The improvements of Matching Networks over baselines show that matching train and test conditions (*i.e.*, special for few-shot learning) leads to significant gains.

Learning from a few examples in machine learning is closely related to the capability of humans to learn concepts and biases. The study in [Feldman, 1997] investigates that few examples suffice to generalize on visual object categorization. For instance, Figure 2.1 shows that generalization is possible with one example. To easily classify the object, we can model the example with deformable parts and geometric configuration, *i.e.*, it consists of one small box on top of the bigger box. Object recognition in extreme condition was studied where provided examples are only few (1 to 5) [Miller et al., 2000; Fei-Fei et al., 2003], at the same time, *one-shot* or *few-shot learning* is coined for the first time for visual object recognition. It is started from the evaluation on handwritten digit recognition in [Miller et al., 2000], where common image deformation is used to improve the performance of the classifiers. Few-shot learning can also be viewed as unsupervised learning [Fei-Fei et al., 2003], where the features are unlabeled and shared representations must be found to group the images to the same class.

2.1.2 Episodic Learning

We start by defining the terminology used in few-shot learning. The task in few-shot learning is in the form of *episode* as shown in Figure 2.2. An episode $\mathcal{T}_i$ for task-i consists of two sets, the support set S and the query set Q. This learning paradigm depicts how machines can improve their ability given fragmented data in each iteration. Specifically, deep embeddings learn with a limited amount of labels and inputs per episode. This learning paradigm is well-known as N-way K-shot classification (*e.g.*, 20-way 1-shot and 5-way 5-shot). We introduce our notations for (N-way, K-shot) few-shot learning. This training strategy mimics the real-world conditions where an agent may only encounter a small number of classes N and a few samples K for a current observation, then making decisions. In contrast to standard training procedure, where the numbers of classes and samples are relatively high because the assumption is that the data is always stored and available after machine learning system deployment. Moreover, the *episodic* style also gives a more generalization formulation to enforce adaptation

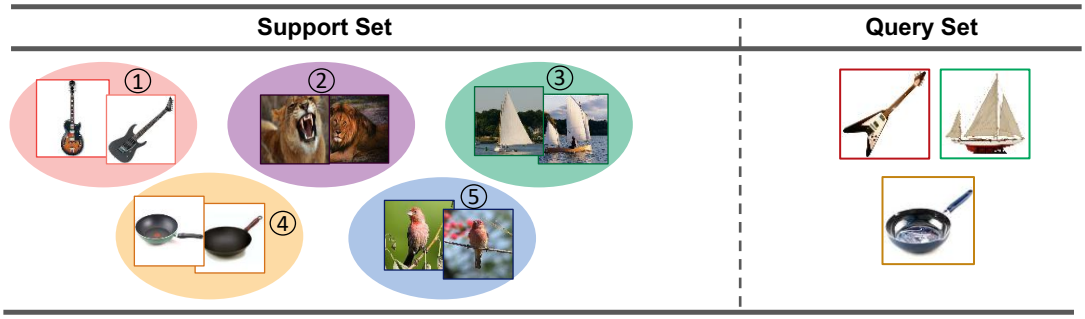


Figure 2.2: An illustration of an *episode*. The support set consists of a few training examples representing some classes and the query set contains instances for performance assessment.

in neural networks. We denote each episode or task as $\mathcal{T}_i$, composed of the support set $S = \{(\boldsymbol{x}_{1,1}, c_{1,1}), (\boldsymbol{x}_{1,2}, c_{1,2}), \cdots, (\boldsymbol{x}_{N,K}, c_{N,K})\}$ and the query set $Q = \{\boldsymbol{q}_1, \cdots, \boldsymbol{q}_{N\times M}\}$, where $\boldsymbol{x}_{i,j}$ is the j-th sample from class i and $c_{i,j} \in \{1, \cdots, N\}$ and there are M query images for each class. To obtain a trained model, *episodes* are used to sample data in a dataset. The reason for using an *episodic* style in training is to match a train and test conditions such that a model learns to recognize a given task.

2.1.3 Few-Shot Learning Methods

DNNs have been very successful in learning discriminative features from images. The works of few-shot learning in [Santoro et al., 2016; Vinyals et al., 2016] attempt to solve few-shot classification with end-to-end deep neural networks. In majority of cases, the network, trained from episodes, aims to infer the underlying discriminative model of specific tasks from limited data. Please see the survey work by [Wang and Yao, 2019] for more details about the settings and state-of-the-arts.

Metric learning based approaches. Few-shot learning based on metric-learning is the closest direction to our work. Matching Nets [Vinyals et al., 2016] and Siamese Nets [Koch et al., 2015] learn to embed inputs by minimizing distance of two different samples, *i.e.*, distances to samples are used to classify the queries. Prototypical Nets [Snell et al., 2017] extends the idea from samples to a class-wise metric. The descriptors from all the samples of a specific class are grouped and considered as the class prototypes. The prototypes are subsequently used for inference. Learning a non-linear relationship between class representations and queries can be modeled by neural networks as shown for example in Relation Nets [Sung et al., 2018]. The underlying metric is learned to preserve small distances between feature vectors sharing the same class label. Qiao et al. [2018] shows that the activation of a network is correlated with weights of its classifier (final layer) and advocates that a prototype made of the activation is sufficient for classification. Furthermore, a neural network as a classifier following new categories can be attached to various tasks such as learning without forgetting or implanting as proposed in [Gidaris and Komodakis, 2018a; Lifchitz et al., 2019]. Other works use feature attention modules [Shi et al., 2019; Fang et al., 2019] to modulate features for few-shot learning [Ye et al.,

2018; Hou et al., 2019a]. The support and query images can also be modeled as parameterized graphs with various distance functions (*e.g.*, Gaussian kernel or linear network), as proposed in [Garcia and Bruna, 2018; Liu et al., 2019d]. Another seminal work [Zhang et al., 2020] uses a differentiable Earth Mover Distance to obtain the optimal matching cost between images.

2.2 Meta-Learning

2.2.1 Overview

Based on a practical definition today, meta-learning refers to *learning to learn*, meaning rapidly improving a current model by learning from a family of similar tasks [Hospedales et al., 2020]. Before delving into details, we recall that the objective in meta-learning is to **1.** achieve rapid convergence for new tasks (task-level) and **2.** to generalize beyond seen tasks (meta-level). To achieve the meta-learning capability, a common approach is to design models that can learn from limited data using the concept of episodic training [Vinyals et al., 2016; Santoro et al., 2016] in which a model is presented with a set of tasks (*e.g.*, image classification), where for each task limited data is available.

To put the discussion into context, we first provide a brief overview of the MAML [Finn et al., 2017] algorithm. Let $\mathcal{D}_\tau^{trn}$ and $\mathcal{D}_\tau^{tst}$ be the training and the validation sets of a given task $\tau \sim p(\mathcal{T})$, respectively. We assume that $\mathcal{D} := \{\boldsymbol{x}_i, y_i\}_{i=1}^{|\mathcal{D}|}$, with an input $\boldsymbol{x}_i \in \mathcal{X}$, and a label $y_i \in \mathcal{Y}$ for some small $|\mathcal{D}|$. Furthermore, let $h : \mathcal{X} \times \mathbb{R}^n \to \mathcal{Y}$ be the functionality of the model of interest, parameterized by $\theta \in \mathbb{R}^n$.

The MAML seeks a universal initialization θ^* by minimizing:

$$\min_{\theta^*} \sum_{\tau \sim P(\mathcal{T})} \mathcal{L}\left(D_\tau^{\text{tst}}, \theta^* - \alpha \sum_{k=0}^{K-1} \nabla \mathcal{L}\left(D_\tau^{\text{trn}}, \theta_\tau^k\right)\right). \tag{2.1}$$

Here, $\theta_\tau^k = \theta_\tau^{k-1} - \alpha \nabla \mathcal{L}\left(D_\tau^{\text{trn}}, \theta_\tau^{k-1}\right)$ with $\theta_\tau^0 = \theta^*$. The loss terms are:

$$\begin{aligned} \mathcal{L}\left(D_\tau^{\text{trn}}, \theta\right) &:= \mathbb{E}_{\boldsymbol{x},y \sim \mathcal{D}_\tau^{\text{trn}}}\left[\ell(h(\boldsymbol{x}, \theta), y)\right], \\ \mathcal{L}\left(D_\tau^{\text{tst}}, \theta\right) &:= \mathbb{E}_{\boldsymbol{x},y \sim \mathcal{D}_\tau^{\text{tst}}}\left[\ell(h(\boldsymbol{x}, \theta), y)\right], \end{aligned} \tag{2.2}$$

where, $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is the loss of the model h. Intuitively, given a task τ, the MAML starts from θ^* and performs K gradient updates on D_τ^{trn} to attain the adapted parameters θ_τ^K (this is called the inner-loop updates). MAML utilizes Stochastic Gradient Descent (SGD) in its inner-loop. Then it uses D_τ^{tst} and θ_τ^K (which is dependent on θ^*) to improve the universal initialization point θ^* (this is called the outer-loop update).

2.2.2 Gradient-base meta-learning

Enjoying some theoretical guarantees, meta-learning by optimization (*e.g.*, [Balcan et al., 2019; Grant et al., 2018]) encompasses methods that learn the parameters of the model, at the meta-level, through gradient updates. The celebrated "model agnostic meta-learning" in [Finn et al., 2017] and "Learning to Learn" in [Andrychowicz et al., 2016] are prime examples of

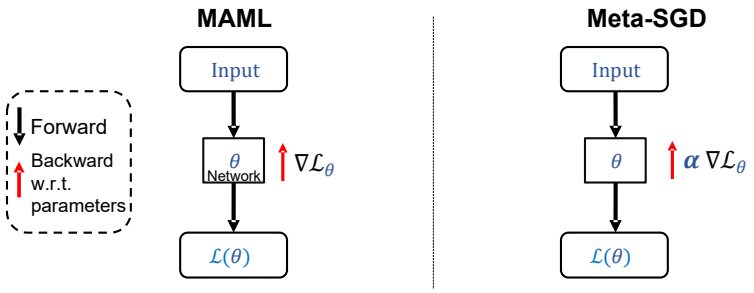


Figure 2.3: Comparison of MAML Finn et al. [2017] and Meta-SGD Li et al. [2017]. Meta-SGD has a modulator to the gradient denoted as $\boldsymbol{\alpha}$.

Method	Inner-Loop Update
MAML [Finn et al., 2017]	$\boldsymbol{\theta}^{(k+1)} \leftarrow \boldsymbol{\theta}^{(k)} - \alpha \nabla_{\boldsymbol{\theta}^{(k)}} \mathcal{L}(\boldsymbol{\theta}^{(k)})$
Meta-SGD [Li et al., 2017]	$\boldsymbol{\theta}^{(k+1)} \leftarrow \boldsymbol{\theta}^{(k)} - \boldsymbol{\alpha} \nabla_{\boldsymbol{\theta}^{(k)}} \mathcal{L}(\boldsymbol{\theta}^{(k)})$
CAVIA [Zintgraf et al., 2019]	$\boldsymbol{\nu}^{(k+1)} \leftarrow \boldsymbol{\nu}^{(k)} - \alpha \nabla_{\boldsymbol{\nu}^{(k)}} \mathcal{L}(\boldsymbol{\pi}, \boldsymbol{\nu}^{(k)}, \boldsymbol{\theta})$
LEO [Rusu et al., 2019]	$\boldsymbol{z}^{(k+1)} \leftarrow \boldsymbol{z}^{(k)} - \alpha \nabla_{\boldsymbol{z}^{(k)}} \mathcal{L}(\boldsymbol{z}^{(k)}, \boldsymbol{\theta}, \boldsymbol{\phi}_e, \boldsymbol{\phi}_d, \boldsymbol{\phi}_r)$

Table 2.2: A comparison of inner-loop updates of various meta-learners. The red font shows additional parameters updated in the outer-loop. In Meta-SGD, $\boldsymbol{\alpha}$ is the modulator to the gradient. In CAVIA, $\boldsymbol{\pi}$ is a parameter used to generate modulation for gradients. In LEO, $\boldsymbol{\phi}_e, \boldsymbol{\phi}_d, \boldsymbol{\phi}_r$ are parameters to generate classifier parameters.

meta-learning by optimization. Note that, our meta-learning approach is an adaptive module acting on the model parameters which differs from relation and metric learning approaches *e.g.*, [Snell et al., 2017; Simon et al., 2020a; Koniusz and Zhang, 2020a].

Extensions of MAML include Reptile [Nichol et al., 2018] that directly combines the updated parameters in the inner loop with that of the meta-learner, leading to a highly scalable meta-learning algorithm. MAML++ [Antoniou et al., 2019] uses a weighted loss in the inner-loop to boost the accuracy. We provide a brief overview on optimization-based meta-learning algorithms (see Table 2.2).

Preconditioned stochastic gradient descent (SGD). The general framework of preconditioning for fast adaptation of a neural network is introduced in [Flennerhag et al., 2020]. We discuss Meta-SGD and Meta-Curvature as the specific forms of preconditioned SGD. In Meta-SGD [Li et al., 2017], the gradient update is equipped with a universal meta-parameter (α). In Meta-Curvature [Park and Oliva, 2019], the meta-parameter has a form of preconditioning matrices which transform the gradient in the inner-loop. During training, the meta-parameters of both methods [Li et al., 2017; Park and Oliva, 2019] change in each meta-update, however, they remain unchanged when the algorithm is applied to unseen tasks (see Fig. 2.3 for a conceptual diagram). Furthermore, additional meta-parameters are required if one requires to use more than one update in the inner-loop while ModGrad does not require extra parameters.

Network modulation and generation. The modulation can be performed in the feature space [Zintgraf et al., 2019]. Borrowing the idea from FiLM [Perez et al., 2018], a function $f_\pi(.)$ is designed to create the modulation for conditioning layers. In the inner-loop,

CAVIA updates only the context vector (ν) which is the input of f_π. The context vector is used as additional inputs to layers of the neural network to facilitate adaptation. In the same spirit, meta-transfer learning [Sun et al., 2019a] learns the functions which scale and shift network parameters for each different task. Another stream of this line of research in meta-learning is by generating neural networks for classifiers and updating the generator's parameters in a low-dimensional space as proposed in [Rusu et al., 2019].

2.3 Incremental Learning

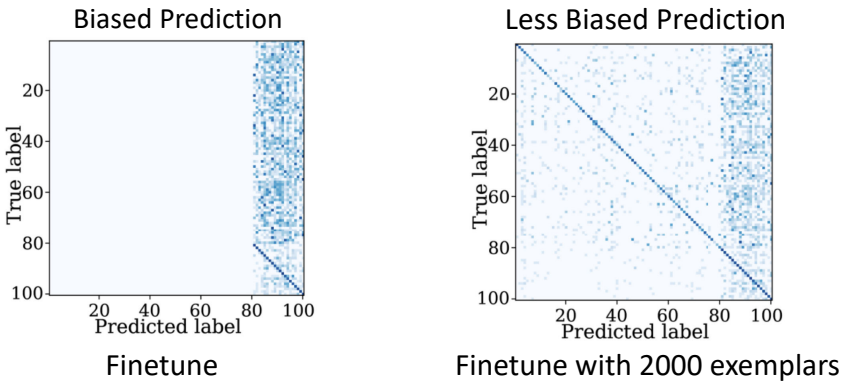


Figure 2.4: The phenomenon of bias prediction when the past data is not replayed in the current task (left) and the less biased prediction by replaying 2000 exemplars (right). This is also known as a core problem in incremental learning called as *catastrophic forgetting*. (Note: The image is taken from [Masana et al., 2020])

2.3.1 Overview

The long-term adaptation in DNNs is formulated as incremental learning in literature, which is under the umbrella of *life-long learning*. Based on the incremental learning in machine learning problems *e.g.*, learning for concept recognition, learning under domain shift, and learning multiple tasks, Van de Ven and Tolias [2019] identify three different scenarios for real-world problems *e.g.*, task incremental learning, domain incremental learning, and class-incremental learning, and this work focuses on class-incremental learning. In detail, task incremental learning is defined as learning tasks sequentially where task IDs are provided, while domain incremental learning has no task IDs and the input distribution changes from one task to another task. In our work, we adopt class-incremental learning where tasks IDs are not provided and new categories are added gradually. The problem appears when learning a new task (*e.g.*, classes 80 to 100) without storing past examples, as we can see in Fig. 2.4 that the predictions are populated around the newly incoming classes. In incremental learning, the problem of forgetting learned information upon learning new information is known as *catastrophic forgetting* [McCloskey and Cohen, 1989; McClelland et al., 1995].

In acquiring new skills, there is so-called *stability-plasticity dilemma*, which is the problem of finding the balance among tasks [Rasch and Born, 2013]. This constraint occurring in human brains and artificial neural systems yields the desiderata in incremental learning, which are to minimize the increase of the model size and maximize the performance gain (backward and forward transfer).

2.3.2 Class incremental learning setting

Below, we define the multi-class IL problem and closely follow the IL setup in [Hou et al., 2019b]. Suppose a sequence of T tasks with associated data $\mathcal{D}_0, \cdots, \mathcal{D}_T$ is given. We denote the data in task t by $\mathcal{D}_t = \{(\boldsymbol{x}_i, y_i)_{i=1}^{N_t}\}$, with $\boldsymbol{x}_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$. In contrast to classical training that has access to every training example, a certain budget is determined for the memory such that only limited amount of examples can be stored $\mathcal{M} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)_{i=1}^{L}\}$. At time $t = 0$, the training data merely consists of $\mathcal{D}_0$. Afterwards, the training data is formed as the union of the data at a given task and the memory (*i.e.*, , $\mathcal{D}_t \cup \mathcal{M}$). Note that, each $\mathcal{D}_t$ for $t > 0$ often has less samples and novel categories (*e.g.*, 2, 5, or 10) as compared to $\mathcal{D}_0$.

In IL, a DNN parameterized with $\boldsymbol{\Phi} = \{\boldsymbol{\theta}, \boldsymbol{\varphi}\}$ is used to realize a mapping in the form $\hat{y} = f(\boldsymbol{x}; \boldsymbol{\Phi})$. Here, $\boldsymbol{\theta}$ represents a part of network that encodes the input into a descriptor as $\boldsymbol{z} = f_{\text{enc}}(\boldsymbol{x}; \boldsymbol{\theta})$. The descriptor is then passed through a classifier parameterized by $\boldsymbol{\varphi}$ to produce the prediction $\hat{y} = f_{\text{cls}}(\boldsymbol{z}, \boldsymbol{\varphi})$. At time t, our goal is to update and improve our model at $\boldsymbol{\Phi}_t$ from the old model $\boldsymbol{\Phi}_{t-1}$ using $\mathcal{D}_t \cup \mathcal{M}$. This, in practice, means that the classifier $\boldsymbol{\varphi}_t$ keeps expanding as more novel classes are observed.

2.3.3 Prior works in class incremental learning

Below, we describe relevant approaches that prevent *catastrophic forgetting* for class-incremental learning.

Representative memories. In IL, the selection of exemplars in the memory is very crucial as we cannot access the whole data from past training phases. The selected exemplars are replayed as representatives from the past categories, as a result, the model does not overfit to the new categories. A method, so-called *herding* [Welling, 2009; Rebuffi et al., 2017b], is used to pick potential exemplars for future use in training models. Rebuffi et al. [2017b] use prototypes (mean of each class) to find the closest samples to be stored in the memory. However, some selected samples may not be optimal for the future training phases, Liu et al. [2020b] propose a meta-learning approach to update the memory by the gradient descent. A memory replay can also be achieved with a generative mechanism as proposed in [Shin et al., 2017; Kemker and Kanan, 2018], but this approach needs additional networks which are not easy to train and optimize. To efficiently keep exemplars in the memory, Iscen et al. [2020] propose a feature adaptation strategy.

Gradient trajectories. A family of these approaches regularize the update direction to be 'less forgetting'. Inspired by the synaptic consolidation from neurobiological models [Fusi et al., 2005], Kirkpatrick et al. [2017] propose Elastic Weight Consolidation (EWC) to regulate the update of DNN using the Fisher Information Matrix and accommodate both the solution from previous task and the new task, thus the training trajectories lead to low errors on both tasks. By

allowing backward transfer along the same direction with the current task, Gradient Episodic Memory (GEM) [Lopez-Paz and Ranzato, 2017] adds a constraint considering the similarity between the gradients of the previous tasks and the gradients of the current task. Riemannian walk [Chaudhry et al., 2018] provides a generalization framework, so-called EWC++, as an extension of EWC and an evaluation method for *forgetting* and *intransigence*, *i.e.*, a model tends to forget learned knowledge and is incapable to learn new tasks, respectively.

Parameter updates. When the models need to face multiple tasks, a common strategy is to utilize a sub-network for a specific tasks. The idea is to use a method similar to dropout Srivastava et al. [2014] to mask the connections between neurons and create a sub-network. Some examples are works in Mallya et al. [2018]; Mallya and Lazebnik [2018] that encourage to use masking or pruning to update selected network parameters for every different task. In addition, Rajasegaran et al. [2019] propose a simple controller with random search to choose the optimal paths for a new task.

Regularization with distillation losses. Training neural networks in an incremental fashion affects the model's capability to maintain the performance of existing categories. An effective strategy is to introduce knowledge distillation [Hinton et al., 2015] between an old model and a new model. A knowledge distillation loss is performed on the output of neural networks as proposed in [Li and Hoiem, 2017] such that it preserves old task performance. To avoid forgetting, Castro et al. [2018] recommend balanced fine-tuning with temporary distillation loss to the classification layers. A distillation loss for incremental learning is extended to the feature space with a measurement using the Euclidean distance proposed in [Hou et al., 2019b; Cheraghian et al., 2021]. However, all of these knowledge distillation methods do not consider the underlying manifold structure between old and current tasks. [Belouadah and Popescu, 2019] argue that knowledge distillation hurts the IL performance when there are at least a few examples. However, our setting in this work follows the work in [Hou et al., 2019b] in which the classifier weight is built based on the notion of cosine distance and not a fully connected layer as in [Belouadah and Popescu, 2019].

2.4 NAS under Label-Noise

2.4.1 Overview

Searching for an optimal neural architecture is of importance for adaptation in a specific task. Adaptation in this context means that neural networks not only update the parameters but also the underlying structure because a new task (*e.g.*, finetuning on a new dataset) might need a novel architecture to achieve optimal performance. However, NAS assumes that the labels have no noise. In a standard training protocol, a neural network suffers from degrading performance when training under label noise [Han et al., 2018; Patrini et al., 2017]. In this setting, we draw attention to search an architecture under label noise. Below, we describe the problems and prior works in the topics of NAS and learning under label noise.

2.4.2 Prior works in neural architecture search

NAS is the process of automating the design of a neural network architecture. NAS has been successfully applied to various recognition and image generation problems [Zoph and Le, 2017; Chen et al., 2019c; Gong et al., 2019]. NAS algorithms can be computationally demanding *e.g.*, when based on evolutionary algorithms [Real et al., 2019; Elsken et al., 2019; Real et al., 2017; Stanley et al., 2019] or reinforcement learning [Zoph and Le, 2017; Baker et al., 2016], which provide flexible schemes to sample and evaluate architectures from a vast pool of architectures (so-called search space). However, searching directly on a very large dataset with a large neural network is computationally expensive. Zoph et al. [2018] propose a scalable solution by searching a cell structure instead of the entire neural network structure. Due to the above issues, our method follows the formulation of differentiable NAS [Liu et al., 2019b; Zela et al., 2020a] which is computationally feasible on a broad range of tasks [Lian et al., 2020; Liu et al., 2019a; Xu et al., 2020] and results in a cell transferrable to other tasks.

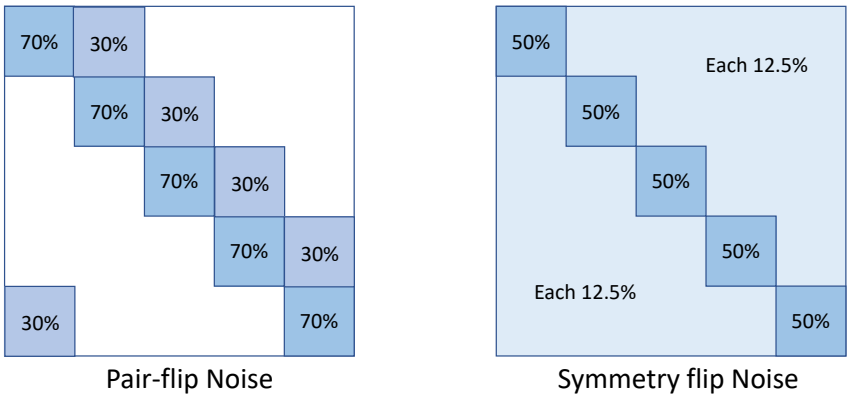


Figure 2.5: An illustration of synthetic noisy labels. There are two types used in our work *e.g.*, pair-flip noise (left) and symmetric noise (right).

2.4.3 Noisy labels

In the setting of our experiments, we use synthetic label noise which is illustrated in Figure 2.5. The labels are flipped with the other labels, then the DNN is trained on this contaminated dataset and evaluated on clean data. There are two types of noisy labels introduced in our setting: pair-flip and symmetry flip. The number of contaminated labels is represented as a percentage in Figure 2.5.

Label noise can harm the performance of neural networks if training is carried out as if data is clean. One of the solutions to tackle label noise is to relabel the samples as proposed in Vahdat [2017]; Tanaka et al. [2018]; Yi and Wu [2019]. Loss correction approaches [Arazo et al., 2019; Reed et al., 2015; Jiang et al., 2018; Zhang and Sabuncu, 2018] achieve the same purpose of addressing label noise by compensating and modifying the loss functions. Several types of noise (*e.g.*, symmetric) can even be modeled as a transition matrix [Patrini et al., 2017; Xia et al., 2019] that indicates the probability of clean labels being flipped to noisy labels. Other

approaches combat noisy labels by ranking [Guo et al., 2018], re-weighting [Ren et al., 2018b] and selecting [Han et al., 2018] samples.

Chapter 3

Subspace Methods for Few-Shot Learning

This chapter introduces a framework for few-shot learning using subspace methods. The aim is to improve the classifiers such that the model can generalize to unseen classes in the low-data regime. The constructed subspace from the support samples acts as a dynamic classifier, and the prediction score for each class is obtained by computing the point-to-subspace distance. In addition, we also propose the subspace-to-subspace distance with kernel methods to improve the performance even further. Our experiment also shows that using subspaces as classifiers for few-shot learning yields robustness when the trained model faces noisy inputs.

3.1 Introduction

Since training DNNs with limited data may lead to unsatisfactory and poor designs, opting for standard models and training protocols with random initialization is an inferior choice. An alternative to learn from a few examples is closely related to the transfer learning methods. The central idea is to first train the DNN with subsets of data points that resemble learning with limited data. The parameters of the DNN are then transferred and adapted for unseen categories with a few examples. This type of learning, known as Few-Shot Learning (FSL), does not only evaluate how the model learns from a few examples but also how well it generalizes in a variety of other previously unseen test categories. Looking at the current progress in FSL, a diverse set of ideas and approaches has been studied, from embedding learning Koch et al. [2015]; Vinyals et al. [2016]; Sung et al. [2018]; Simon et al. [2020a]; Zhang et al. [2020]; Ma et al. [2021]; Zhang et al. [2021], to adaptation techniques Finn et al. [2017]; Ravi and Larochelle [2017]; Rusu et al. [2019]; Park and Oliva [2019]; Flennerhag et al. [2020]; Simon et al. [2020b]; Liu et al. [2020a].

Consider designing a Convolutional Neural Network (CNN) for an image classification problem. A standard CNN consists of convolutional layers as a feature extractor and Fully-Connected (FC) layers as a classifier. In FSL, the training and testing class concepts are disjoint. Thus, a fixed classifier (*e.g.*, FC layers) has no ability to generalize to unseen class concepts. In this chapter, we firstly formulate FSL as a two-stage learning process which **1.** learns a generic feature extractor followed by **2.** generating a classifier dynamically from some limited data. Many state-of-the-art FSL techniques can be extended to such a learning paradigm.

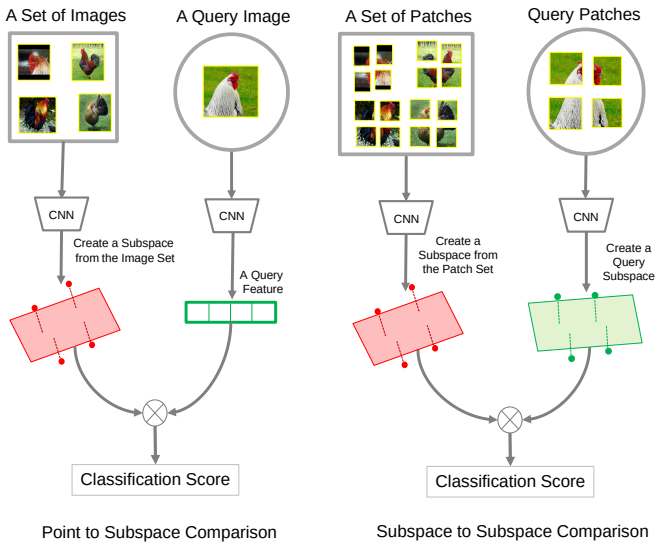
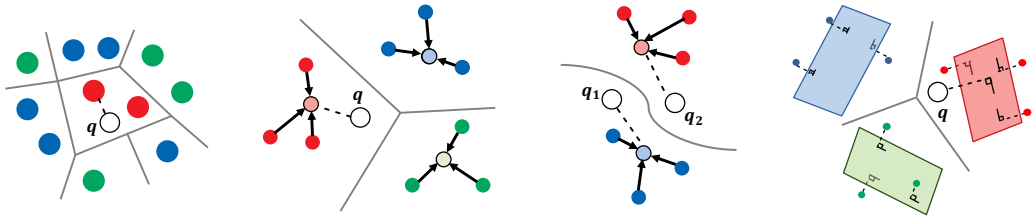


Figure 3.1: The overview of our FSL approach. Left: A subspace is created to embed the set of image features from the support set. The comparison is based on the distance of a query feature (point) to the subspace. Right: Two subspaces are created from the image patches of the support set and the query patches, and the distance is evaluated between these two subspaces.

Our main focus in this chapter is how one can reliably generate a classifier from the limited data. Aside from limited annotations, a requirement in many challenging FSL problems is to learn the classifier from high-dimensional data points. This ultimately requires learning a symmetric function[1] from some high-dimensional data. For instance, concatenating features before feeding to an FC layer is not symmetric and might yield different outcomes if the order changes. We make a contribution and propose to construct the symmetric function using subspaces which have a long history in modeling visual data Turk and Pentland [1991a]; Basri and Jacobs [2003]; Zhou et al. [2018]; Wang and Cherian [2019]. Our idea differs from previous studies such as Snell et al. [2017], where the symmetric function is realized through a form of average pooling.

As illustrated in Fig. 3.1, we propose two approaches for applying subspace methods for FSL. In the first approach, we create subspaces for every class using all available images of the class. We then employ class-specific subspaces to measure the likelihood of query samples. We empirically observe that such modeling will improve the classification performance and will show that the resulting algorithm is more robust to various perturbations in comparison to a baseline method. On the downside, constructing class-specific subspaces directly from images cannot be used to address 1-shot problems without alternation. This is simply because to construct a subspace, one needs more than one sample. To overcome this limitation and as our second contribution, we propose an FSL method that exploits dissimilarity between subspaces as a means for learning from limited data. Fig. 3.1 (right) illustrates how our subspace method can be employed when a bag of features per image is available (*e.g.*, by splitting an image into

[1] A symmetric function is invariant to any permutation of its input arguments (*e.g.*, features).



(a) Matching Nets (b) Proto Nets (c) Relation Nets (d) DSN

Figure 3.2: Various classifiers for few-shot classification. (a) Matching Nets create pairwise classifiers. (b) Prototypical Nets create mean classifiers based on samples in the same class. (c) Relation Nets produce non-linear classifiers. (d) Our proposed method creates classifiers using the notion of subspace-to-point distance.

patches). One fundamental difference between the two algorithms is the similarity function used to measure the class likelihoods, the former method employs a point-to-subspace distance while the latter opts for subspace-to-subspace distances, respectively.

We further develop a kernelized version of our subspace to subspace approach. This, aside from its theoretical properties (*e.g.*, the resulting method can be applied to non-vectorial data), can boost the performances by making use of subspaces in infinite-dimensional spaces. We also study the effect of fast approximations to subspace creation and investigate the denoising capacity of the proposed subspace method.

3.2 Subspace Methods

3.2.1 Dynamic Classifier

Our proposed model has two main modules, namely a feature extractor and the dynamic classifier. Let $f_{\boldsymbol{\theta}} : \mathcal{X} \rightarrow \mathcal{Z}$ be a mapping from the input space $\mathcal{X}$ to a D-dimensional latent space realized by a neural network. Furthermore, let $f_w : \mathcal{Z} \rightarrow \mathcal{Y}$ be a dynamic classifier that maps the D-dimensional latent vectors to their associated predictions. We formulate the problem of FSL as generating dynamic classifiers according to:

$$\begin{aligned} \min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \quad & J(\boldsymbol{\theta}, \boldsymbol{W}, \mathcal{D}^{\mathtt{tst}}), \\ \text{where} \quad & \boldsymbol{W} = g(\boldsymbol{\theta}, \mathcal{D}^{\mathtt{trn}}). \end{aligned} \tag{3.1}$$

In our formulation, $g(\cdot)$ is a function that realizes the parameters of the classifier (*i.e.*, , $\boldsymbol{W}$) given $\mathcal{D}^{\mathtt{trn}}$. As an example, $g(\cdot)$ can be a mean aggregation function among samples within a class, as in [Snell et al., 2017]. We can interpret the above equation as generating a new classifier based on a specific context (given by $\mathcal{D}^{\mathtt{trn}}$). Specifically, $J(.)$ can be a cross-entropy loss evaluated on the validation data.

Why do we need to model FSL using dynamic classifiers? The problem of FSL is dynamic in nature. This is not only because samples can be assigned to different classes across episodes (in episodic learning, this is helpful to achieve better generalization), but also because the model

may face unseen classes. That said, we can safely assume that the feature extractor should remain unchanged (as it is responsible for generating rich universal feature representations). However, the same cannot be said about the classifier, and we need to adapt it to match the context in an episode. To formulate this aspect, we can measure the likelihood of a class c given a query sample $\boldsymbol{q}$ with a softmax layer as:

$$p(c \mid \boldsymbol{q}) = \frac{\exp(\boldsymbol{w}_c^\top f_\theta(\boldsymbol{q}))}{\sum_{c'} \exp(\boldsymbol{w}_{c'}^\top f_\theta(\boldsymbol{q}))}, \tag{3.2}$$

where $\boldsymbol{w}_c$ is the weight vector of class c. Then, the problem of FSL becomes a task of how $\boldsymbol{w}_c$ for $c \in \{1, 2, \cdots, N\}$ can be generated once a new task arrives. To showcase this setup in a prior FSL work, we discuss the pair-wise, the prototype, and the non-linear classifier below.

Pair-wise Classifier. It is possible to build a classifier directly from samples by calculating the similarity between them as shown in Fig. 3.2 (a). One seminal work in this direction is the Matching Nets [Vinyals et al., 2016]. The samples are embedded via LSTMs and attention modules. However, this method does not impose an invariance w.r.t. the order of input images per episode, which may affect the accuracy.

Prototype Classifier. Qian *et al*. observe that there is a strong correlation between the parameters of a classifier and class prototypes [Qiao et al., 2018]. This suggests that in FSL, a classifier can be generated via class prototypes. The work of Snell *et al*. shows that learning embeddings along with feature averaging per class can lead to rich prototypes for FSL [Snell et al., 2017]. More specifically, the $\boldsymbol{w}_c$ is defined as:

$$\boldsymbol{w}_c = \frac{1}{K} \sum_{i:y_i=c} f_\theta(\boldsymbol{x}_i) \ ,$$

which preserves the symmetric property (invariance to order of images in episode) because the averaging is order-agnostic (see Fig. 3.2 (b)). Some of the following works, including [Wang et al., 2018; Gidaris and Komodakis, 2018b] can also be interpreted as designing dynamic classifiers on the fly.

Relational Classifier. Instead of relying on pairwise similarities (as done in [Sung et al., 2018]), one can opt for relative similarities as a means for learning. For example, Relational Nets use a non-linear binary classifier to calculate the similarity (see Fig 3.2 (c) for a conceptual illustration). Let $\mathbb{R}^{2D} \ni \boldsymbol{h}_i = [f_\theta(\boldsymbol{x}_i); f_\theta(\boldsymbol{q})]$ denote the concatenation of $f_\theta(\boldsymbol{x}_i)$ and $f_\theta(\boldsymbol{q})$. The learning can now be formulated as a binary classification problem by identifying $\boldsymbol{w} \in \mathbb{R}^{2D}$ to enforce $\pi(\boldsymbol{h}_i^\top \boldsymbol{w})$ to be high for samples from the same class, while being low for disjoint labels. Here, π is a non-linear function, typically monotonic (*e.g.*, obtained from the inverse link of a proper loss, like the sigmoid).

3.2.2 Subspaces for Few-Shot Classification

In this chapter, we model samples in the support set using subspaces, which inherently yields lower-dimensional manifolds for data representation. One well-established technique to attain low dimensional representation is the Principal Component Analysis (PCA). In practice, PCA

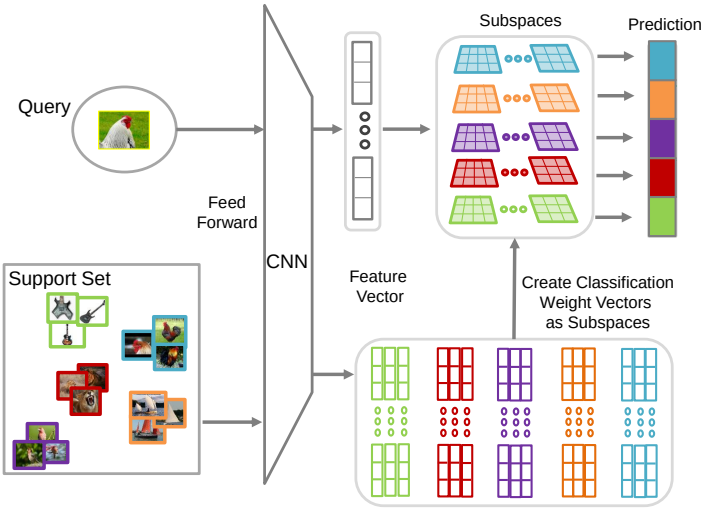


Figure 3.3: The overall pipeline of our approach. The subspace classifier replaces a classifier with a single vector per class. A discriminative objective is then applied to maximize the margin between subspaces.

has been used to address a broad range of problems in computer vision, including the pioneer works on feature selection [Fukunaga and Koontz, 1970], object detection [Murase and Nayar, 1997], and face recognition [Turk and Pentland, 1991b]. Add to this the fact that PCA enjoys several unique properties including optimality in compression (w.r.t. l_2 measure). Moreover, PCA can be used to reduce the effect of noise, and to combat missing data [Cunningham and Ghahramani, 2015].That said, we note that PCA also comes with a number of limitations, most importantly, the model is linear and its unique properties follow (in many cases) if the distribution of data can be assumed to be Gaussian. PCA models data as a linear subspace spanned by basis vectors $\boldsymbol{P} = [\boldsymbol{u}_1, \cdots, \boldsymbol{u}_n]; \boldsymbol{P} \in \mathbb{R}^{D \times n}$. It can be shown that the principal components (*i.e.*, , $\boldsymbol{u}_i$) are the solution to (see [Cunningham and Ghahramani, 2015])[2]

$$\begin{aligned} \min_{\boldsymbol{P}} \ & \left\| \boldsymbol{Z} - \boldsymbol{P}\boldsymbol{P}^\top \tilde{\boldsymbol{Z}} \right\|_F^2 \\ \text{s.t.} \ & \boldsymbol{P}^\top \boldsymbol{P} = \mathbf{I}, \end{aligned} \tag{3.3}$$

where $\tilde{\boldsymbol{Z}} \in \mathbb{R}^{D \times K}$ denote a matrix consisting of feature vectors. For FSL, we propose to model points by subspaces for each class. That is, we will use a set of N subspaces $\{\boldsymbol{P}_i\}_{i=1}^N$, each associated to a class to perform FSL. Our goal is to learn the feature extractor (*i.e.*, , f_Θ) to subsequently generate subspaces, hence the feature extractor should yield representations that in a way are suitable for forming subspace classifiers.

[2]The basis for a subspace can be obtained by Singular Value Decomposition (SVD). We emphasize that more involved techniques to obtain robust subspaces can potentially improve the algorithm. Nevertheless, our goal is to assess whether the concept of subspace modeling for few-shot learning is well justified, thus we opt for the truncated SVD in our implementation.

3.2.3 Subspace Classifier

One can imagine that higher-order statistics can be beneficial for classification [Cherian, 2013; Harandi et al., 2014; Maji et al., 2012; Koniusz et al., 2016]. In our work, we model a classifier as a projection realized by a subspace in the form $\boldsymbol{P}\boldsymbol{P}^\top$, which effectively makes use of higher-order statistics to encourage robustness. Below, we describe how to create a subspace and perform classification based on it. A new set of K samples encoded by $f_{\boldsymbol{\theta}}$ (the non-linear feature extraction module) can be expressed as $\boldsymbol{Z}_c = [f_{\boldsymbol{\theta}}(\boldsymbol{x}_{c,1}) - \boldsymbol{\mu}_c, \cdots, f_{\boldsymbol{\theta}}(\boldsymbol{x}_{c,K}) - \boldsymbol{\mu}_c]$, where $\boldsymbol{\mu}_c = \frac{1}{K}\sum_{i:y_i=c} f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$. Our idea for classification using a subspace is simply to find the closest distance between data points to their projections onto the subspace. To this end, a class-specific projection matrix $\boldsymbol{P}_c$ is calculated from $\boldsymbol{Z}_c$ (*e.g.*, using SVD). Now a query latent vector $\boldsymbol{z}_q = f_{\boldsymbol{\theta}}(\boldsymbol{q})$ can be projected onto $\boldsymbol{P}_c$, and classification based on the shortest distance between the query and class-specific subspaces (in the ambient space) is performed. To be specific, our general subspace classifier uses the following point to subspace distance:

$$d^2(\boldsymbol{q}, \boldsymbol{P}_c) = \left\| (\mathbf{I} - \boldsymbol{P}_c\boldsymbol{P}_c^\top)(\boldsymbol{z}_q - \boldsymbol{\mu}_c) \right\|_2^2 . \tag{3.4}$$

This yields the distance of a query point $\boldsymbol{z}_q$ to a class-specific subspace $\boldsymbol{P}_c$.We define the probability of the query being assigned to class c using a softmax function as:

$$p(c \mid \boldsymbol{q}) = \frac{\exp\big(-d^2(\boldsymbol{q}, \boldsymbol{P}_c)\big)}{\sum_{c'} \exp\big(-d^2(\boldsymbol{q}, \boldsymbol{P}_{c'})\big)}. \tag{3.5}$$

A useful observation is that the squared norm in (3.4) can be simplified to inner products and as such in (3.5), one only needs to compute terms in the form $\langle \boldsymbol{P}_c^\top \boldsymbol{z}_q, \boldsymbol{P}_c^\top \boldsymbol{\mu}_c \rangle = \boldsymbol{z}_q^\top \boldsymbol{P}_c \boldsymbol{P}_c^\top \boldsymbol{\mu}_c, \forall c$. Now, we can minimize the negative log of (3.5) and update parameters $\boldsymbol{\theta}$. To train the whole framework, backpropagation through SVD is required, which is readily available in deep learning packages such as PyTorch [Paszke et al., 2017]. Hereafter, we call our proposed method as Deep Subspace Networks (DSN). Our overall pipeline is depicted in Fig. 3.3. The loss for a sample $\boldsymbol{q}$ in DSN is defined as:

$$\ell(\boldsymbol{q}, \mathcal{D}^{\mathtt{trn}}) = d^2(\boldsymbol{q}, \boldsymbol{P}_{c=y_i}) + \log\Big(\sum_{c'} \exp\big(-d^2(\boldsymbol{q}, \boldsymbol{P}_{c'})\big)\Big). \tag{3.6}$$

3.2.4 Discriminative Deep Subspace Networks

In this section, we describe how to benefit from the geometry of subspaces and Grassmann manifolds [Edelman et al., 1998] to improve the DSN. Our goal is to enforce the class-specific subspaces to be more discriminative. In doing so, we need a way to measure the similarity between subspaces. Before explaining how this is done, let us formally define some concepts related to Grassmannian.

Definition 1 (Grassmann Manifold)**.** *The set of n-dimensional linear subspaces of $\mathbb{R}^D, 0 < n < D$, forms the Grassmann manifold $\mathcal{G}(n, D)$. An element $\boldsymbol{P}$ of $\mathcal{G}(n, D)$ can be specified by a $D \times n$ matrix with orthonormal columns* (i.e., *, $\mathcal{G}(n, D) \ni \boldsymbol{P} \Rightarrow \boldsymbol{P}^\top \boldsymbol{P} = \mathbf{I}_n$).*

Algorithm 1 Train Deep Subspace Networks

Input: Episodes $\mathcal{T}_i$ with $\mathcal{D}^{\texttt{trn}}$ and $\mathcal{D}^{\texttt{tst}}$

$\Theta \leftarrow$ random initialization
for t in $\{\mathcal{T}_1, ..., \mathcal{T}_{N_\mathcal{T}}\}$ **do**
  for c in $\{1, ..., N\}$ **do**
    $\boldsymbol{X}_c \leftarrow \{\boldsymbol{x}_i \in \mathcal{D}^{\texttt{trn}} | y_i = c\}$
    Obtain latent vectors $\tilde{\boldsymbol{Z}}_c = \boldsymbol{Z}_c - \boldsymbol{\mu}_c \boldsymbol{1}^\top$
    $[\boldsymbol{U}_c, \boldsymbol{\Lambda}_c, \boldsymbol{V}_c^\top] \leftarrow$ decompose$\tilde{\boldsymbol{Z}}_c$
    $\boldsymbol{P}_c \leftarrow n$ leading principal comp. $\boldsymbol{U}_c$
    for $\boldsymbol{q}$ in $\mathcal{D}^{\texttt{tst}}$ **do**
      Compute $d^2(\boldsymbol{q}, \boldsymbol{P}_c)$ using Eq. (3.4)
    end for
  end for
  Compute final loss $\mathcal{L}_t$ using Eq. (3.8)
  Update Θ using $\nabla \mathcal{L}_t$
end for

The notion of similarity on Grassmannian is defined by the minimal distance between points. In our work, we will use the projection distance $\delta : \mathcal{G}(n, D) \times \mathcal{G}(n, D) \rightarrow \mathbb{R}^+$ defined as [Hamm and Lee, 2008]

$$\delta^2(\boldsymbol{P}_i, \boldsymbol{P}_j) = \left\| \boldsymbol{P}_i \boldsymbol{P}_i^\top - \boldsymbol{P}_j \boldsymbol{P}_j^\top \right\|_F^2 = 2n - 2 \left\| \boldsymbol{P}_i^\top \boldsymbol{P}_j \right\|_F^2. \tag{3.7}$$

Now to encourage the class-specific subspaces to be more discriminative, we can indeed maximize the pairwise distances between subspaces, or equivalently minimizing $\|\boldsymbol{P}_i^\top \boldsymbol{P}_j\|_F^2$ for all subspaces. Putting things together, we propose the following loss to learn the parameters of our network:

$$d^2(\boldsymbol{q}, \boldsymbol{P}_{c=y_i}) + \log \Big(\sum_{c'} \exp \big(- d^2(\boldsymbol{q}, \boldsymbol{P}_{c'}) \big) \Big) + \lambda \sum_{i \neq j} \left\| \boldsymbol{P}_i^\top \boldsymbol{P}_j \right\|_F^2. \tag{3.8}$$

The second term in (3.8) is the distance between all pairs of subspaces $i \neq j$, which we maximize. Algorithm 1 explains the steps of training with DSN.

3.2.5 Subspace-to-Subspace Networks

Our initial method which uses the point-to-subspace distance does not lend itself to sets of features per the query image. Thus, in what follows, we extend our idea to cases where each sample is described by a bag of features (*e.g.*, a feature set). Our approach is inspired by the strategy using a bag of features employed in Zhang et al. [2020] to further improve the overall accuracy. We assume that each sample (*e.g.*, a query) is described by a bag of H features. Therefore, the support set and a query sample are represented by $\boldsymbol{Z}_s \in \mathbb{R}^{D \times HNK}$ and $\boldsymbol{Z}_q \in \mathbb{R}^{D \times H}$, respectively. There are several methods to forming bag of features, for which some examples are discussed in [Zhang et al., 2020]. We can pass an image to a CNN and

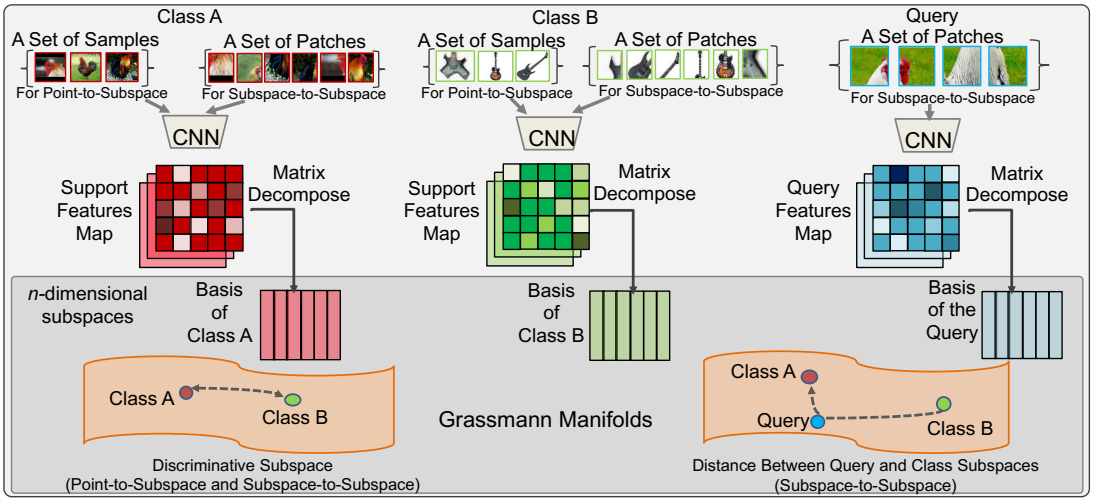


Figure 3.4: Different uses of the Grassmann manifolds in our FSL variants. On the left, we illustrate the discriminative term which maximizes the subspace-to-subspace distance to separate subspaces, each representing a different class (see the right part of Eq. (3.8)). On the right, we show that we minimize the distance between the query and support sets with the use of subspace-to-subspace distance (see Eq. (3.9)). The color-coded points are Grassmann points representing linear-n dimensional subspaces.

form the bag of features from the output with H regions instead of pooling over these regions. Another strategy is to crop H patches from an image at random and obtain their embeddings.

The objective in the subspace-to-subspace approach is to find the minimum distance between the query and class-specific subspaces. The probability of a query being assigned to class c can be defined as:

$$p_{c,q} = p(c \mid \boldsymbol{q}) = \frac{\exp\left(-\delta^2(\boldsymbol{P}_c, \boldsymbol{P}_q)\right)}{\sum_{c'} \exp\left(-\delta^2(\boldsymbol{P}_{c'}, \boldsymbol{P}_q)\right)}, \tag{3.9}$$

where $\boldsymbol{P}_q$ is a matrix containing left-singular vectors constructed from the decomposition of $\tilde{\boldsymbol{Z}}_q = \boldsymbol{Z}_q - \boldsymbol{\mu}_q \mathbf{1}^\top$ for query $\boldsymbol{q}$. We note that $\boldsymbol{P}_c$ is attained from all features in class c.

3.2.6 Kernelized Subspaces on Grassmann Manifolds

We will take a further step in making use of subspaces and develop a kernel extension of DSN, thus enriching our algorithm with infinite-dimensional Grassmannians. Let $k : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ be a positive definite kernel with $\phi : \mathbb{R}^D \to \mathcal{H}$ being its associated implicit mapping to a Reproducing Kernel Hilbert Space (RKHS). We recall that the kernel function measures the inner products in $\mathcal{H}$, *i.e.*, , $k(\tilde{\boldsymbol{z}}, \tilde{\boldsymbol{z}}') = \langle \phi(\tilde{\boldsymbol{z}}), \phi(\tilde{\boldsymbol{z}}') \rangle = \phi(\tilde{\boldsymbol{z}})^\top \phi(\tilde{\boldsymbol{z}}')$. Let $\boldsymbol{\Phi}_c = [\phi(\tilde{\boldsymbol{z}}_1), \cdots, \phi(\tilde{\boldsymbol{z}}_{HK})]$ where $\tilde{\boldsymbol{z}} \in \tilde{\boldsymbol{Z}}_c$ and $\boldsymbol{\Phi}_q = [\phi(\tilde{\boldsymbol{z}}_{q_1}), \cdots, \phi(\tilde{\boldsymbol{z}}_{q_H})]$ be a matrix with features from class c and features from a query q. The gram matrices and their decomposition for a class-wise support and query feature sets are:

$$\boldsymbol{\Phi}_c^\top \boldsymbol{\Phi}_c = \boldsymbol{U}_c \boldsymbol{\Lambda}_c \boldsymbol{U}_c^\top \quad \text{and} \quad \boldsymbol{\Phi}_q^\top \boldsymbol{\Phi}_q = \boldsymbol{U}_q \boldsymbol{\Lambda}_q \boldsymbol{U}_q^\top. \tag{3.10}$$

The subspaces spanning class c and the query are:

$$\boldsymbol{\Psi}_c = \boldsymbol{\Phi}_c \boldsymbol{U}_c \boldsymbol{\Lambda}_c^{-1/2} \text{ and } \boldsymbol{\Psi}_q = \boldsymbol{\Phi}_q \boldsymbol{U}_q \boldsymbol{\Lambda}_q^{-1/2}. \tag{3.11}$$

The inner product between two subspaces can be formulated as:

$$\begin{aligned} \langle \boldsymbol{\Psi}_q, \boldsymbol{\Psi}_c \rangle &= \text{Tr}(\boldsymbol{\Psi}_q^\top \boldsymbol{\Psi}_c) \\ &= \text{Tr}(\boldsymbol{\Lambda}_q^{-1/2} \boldsymbol{U}_q^\top \boldsymbol{\Phi}_q^\top \boldsymbol{\Phi}_c \boldsymbol{U}_c \boldsymbol{\Lambda}_c^{-1/2}) \\ &= \text{Tr}(\boldsymbol{\Lambda}_q^{-1/2} \boldsymbol{U}_q^\top \mathcal{K}(\tilde{\boldsymbol{Z}}_q, \tilde{\boldsymbol{Z}}_c) \boldsymbol{U}_c \boldsymbol{\Lambda}_c^{-1/2}), \end{aligned} \tag{3.12}$$

where $\mathcal{K}(\cdot, \cdot)$ is a kernel matrix between a query and features within a class c. With the inner product between subspaces at our disposal, we can readily reformulate the subspace to subspace solution to benefit from the rich RKHS $\mathcal{H}$. Note that in the FSL problem, the number of samples or patches is small thus the dimensionality of kernel matrices in (3.10) and (3.12) is significantly lower than the dimensionality D of features, which provides a complementary way of dealing with subspaces in FSL. Algorithm 2 shows how we apply our kernelized subspaces on Grassmann manifolds to FSL.

Kernel Choices. The only condition in (3.12), is the availability of a valid positive definite kernel function. There are several kernel functions in the literature that can be used to form a kernel matrix. For brevity, in our evaluations, we opt for a "baseline" linear kernel $k(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^\top \boldsymbol{y}$ and a Radial Basis Function (RBF) kernel $k(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\gamma \|\boldsymbol{x} - \boldsymbol{y})\|_2^2)$.

Algorithm 2 Kernelized Subspace-to-Subspace FSL

Input: Episodes $\mathcal{T}_i$ with $\mathcal{D}^{\texttt{trn}}$ and $\mathcal{D}^{\texttt{tst}}$

for t in $\{\mathcal{T}_1, ..., \mathcal{T}_{N_\mathcal{T}}\}$ **do**
  for c in $\{1, ..., N\}$ **do**
    Get image patches $[\boldsymbol{x}_1, \cdots, \boldsymbol{x}_{HK}]$ from $\boldsymbol{X}_c$
    Obtain latent vectors $\tilde{\boldsymbol{Z}}_c = [\tilde{\boldsymbol{z}}_1, \cdots, \tilde{\boldsymbol{z}}_{HK}]$
    for $\boldsymbol{q}$ in $\mathcal{D}^{\texttt{tst}}$ **do**
      Get image patches $[\boldsymbol{q}_1, \cdots, \boldsymbol{q}_H]$
      Obtain latent vectors $\tilde{\boldsymbol{Z}}_q = [\tilde{\boldsymbol{z}}_{q_1}, \cdots, z\tilde{\boldsymbol{z}}_{q_H}]$
      Compute kernel matrices in Eq. (3.10)
      $\boldsymbol{U}_c, \boldsymbol{U}_q, \boldsymbol{\Lambda}_c, \boldsymbol{\Lambda}_q \leftarrow$ decompose $\boldsymbol{\Phi}_c^\top \boldsymbol{\Phi}_c$, $\boldsymbol{\Phi}_q^\top \boldsymbol{\Phi}_q$
      Compute $\langle \boldsymbol{\Psi}_q, \boldsymbol{\Psi}_c \rangle$ using Eq. (3.12)
    end for
    Compute predictions based on the prob. in Eq. (3.9)
  end for
end for

3.2.7 Fast Subspace Creation

There are several methods commonly used to create subspaces. As described previously, our method uses SVD to form subspaces. However, SVD is computationally costly, especially

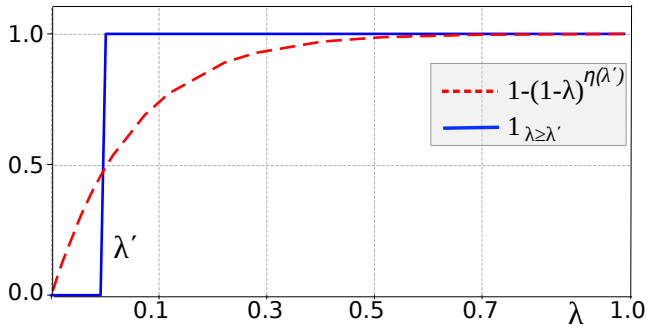


Figure 3.5: The push-forward functions for a cut-off λ' mapping the spectrum of a covariance matrix to the spectrum of the Grassmann feature map $\boldsymbol{P}\boldsymbol{P}^\top$ and the spectrum of MaxExp(F), respectively. See text for details.

during the backpropagation step, and unstable for ill-conditioned cases. The outer product of a truncated projection matrix can be approximated by whitening of $\boldsymbol{\Lambda}$. To this end, we employ a computationally fast spectral operator MaxExp(F) [Koniusz and Zhang, 2020b] which (i) can be realized by mere matrix-matrix multiplications and (ii) enjoys a sub-linear complexity w.r.t. the integer parameter $\eta \geq 1$ controlling the level of whitening:

$$\boldsymbol{P}_c\boldsymbol{P}_c^\top \approx \mathbf{I} - (\mathbf{I} - \boldsymbol{G}_c)^\eta, \tag{3.13}$$

where $\boldsymbol{G}_c = \tilde{\boldsymbol{Z}}_c\tilde{\boldsymbol{Z}}_c^\top / \mathrm{Tr}(\tilde{\boldsymbol{Z}}_c\tilde{\boldsymbol{Z}}_c^\top)$ is the covariance matrix. Proposition 1 illustrates that the above approximation holds. In the FSL setting, we have a query latent vector $\tilde{\boldsymbol{z}}_q = \boldsymbol{z}_q - \boldsymbol{\mu}_c$ that needs to be projected onto the subspace. Thus, by substituting (3.13) into (3.4), we readily obtain:

$$d^2(\boldsymbol{q}, \boldsymbol{P}_c) \approx \left\| \tilde{\boldsymbol{z}}_q - \left(\mathbf{I} - (\mathbf{I} - \boldsymbol{G}_c)^\eta\right)\tilde{\boldsymbol{z}}_q \right\|_2^2 = \left\| (\mathbf{I} - \boldsymbol{G}_c)^\eta \, \tilde{\boldsymbol{z}}_q \right\|_2^2. \tag{3.14}$$

Let $\overline{\boldsymbol{G}}_c = \mathbf{I} - \boldsymbol{G}_c$, now $\overline{\boldsymbol{G}}_c^\eta$ can be computed with a sub-linear complexity w.r.t. the integer $\eta \geq 1$ by matrix exponentiation by squaring [Koniusz and Zhang, 2020b]. The above formulation enjoys $\mathcal{O}(m \log \eta)$ complexity, where m is the cost of matrix-matrix multiplication. For $D \times D$ dimensional matrix, $m \sim D^3$ but this complexity, in theory, can be reduced on the GPU even down to $m \sim \log(D)$ with a heavy parallelization. The above approach can be readily used especially if the discriminative term from Eq. (3.7) needs to be also evaluated. Then, the distance between subspaces from Eq. (3.7) simplifies to:

$$\begin{aligned} \delta^2\left(\boldsymbol{P}_i, \boldsymbol{P}_j\right) &\approx \left\| \left(\mathbf{I} - (\mathbf{I} - \boldsymbol{G}_i)^\eta\right) - \left(\mathbf{I} - (\mathbf{I} - \boldsymbol{G}_j)^\eta\right) \right\|_F^2 \\ &= \left\| (\mathbf{I} - \boldsymbol{G}_i)^\eta - (\mathbf{I} - \boldsymbol{G}_j)^\eta \right\|_F^2. \end{aligned} \tag{3.15}$$

However, if the discriminative term is not required, Eq. (3.14) can be evaluated via matrix-vector

multiplications as follows:

$$d^2(\boldsymbol{q}, \boldsymbol{P}_c) \approx \|\boldsymbol{v}^{(\eta)}\|_2^2 \text{ where } \boldsymbol{v}^{(t+1)} = \boldsymbol{v}^{(t)} - \tilde{\boldsymbol{Z}}_c(\tilde{\boldsymbol{Z}}_c^\top \boldsymbol{v}^{(t)})/\tau, \tag{3.16}$$

and $\tau = \text{Tr}(\tilde{\boldsymbol{Z}}_c\tilde{\boldsymbol{Z}}_c^\top) = \|\tilde{\boldsymbol{Z}}_c\|_2^2$, with $\boldsymbol{v}^{(0)} = \tilde{\boldsymbol{z}}_q$. Eq. (3.16) is equivalent to Eq. (3.14) but with the complexity $\mathcal{O}(m'\eta)$, where $m' \sim k'D$ concerns the cost of the matrix-vector multiplication, which may be further sped-up (by heavy parallelization on the GPU) such that $m' \sim \log k' + \log D$. Constant $k' = K$ (*i.e.*, , number of available images in a class) or $k' = H$ (*i.e.*, , the number of features per image) depending on the variant of our approach. As is clear, the choice between (3.14) and (3.16) depends on the available cost of matrix-matrix and matrix-vector multiplications, respectively. The integer η controls the quality of approximation, and it coincides with $\log \eta$ iterations required to obtain the approximation for Eq. (3.14) and (3.15). The above number of iterations stems from the matrix exponentiation by squaring (see Alg. 2 [Koniusz and Zhang, 2020b]). Moreover, integer η coincides with η iterations for Eq. (3.16) due to the loop computing the recursion in Eq. (3.16).

Proposition 1. *For* $\eta \to \infty$ *and* $0 < \lambda \leq 1$ *(the spectrum is trace-normalized),* $\mathbf{I} - (\mathbf{I} - \boldsymbol{G}_c)^\eta \to \mathbf{I}$. *This result shows that for a sufficiently large integer* $\eta > 1$, *Eq.* (3.13) *performs spectral whitening just as Grassmann feature maps do by design, except for their spectral cut-off that zeroes the lowest eigenvalues, by choice. For instance, the spectrum of the Grassmann feature map can be described as* $1_{\lambda \geq \lambda'}$ *if* $\lambda \geq \lambda'$ *and* 0 *otherwise. By analogy, for* $\eta(\lambda') = \log(0.5)/\log(1-\lambda')$, *one obtains a soft spectral cut-off in Eq.* (3.13) *for eigenvalues below* λ' *because* $1 - (1-\lambda')^{\eta(\lambda')} = 0.5$, $1 - (1-\lambda)^{\eta(\lambda')} \to 0$ *if* $\lambda \to 0$, *and it rapidly tends to 1 if* $\lambda > 0$.

Proof. The first part of the proposition holds as $\lim_{\eta\to\infty} 1 - (1-\lambda)^\eta = 1$, thus we have $\boldsymbol{U}\boldsymbol{U}^\top = \mathbf{I}$, where $\boldsymbol{U}$ are eigenvalues of $\boldsymbol{G}_c$. For the second part, we simply set $1 - (1-\lambda')^\eta = 0.5$ and solve for η. We notice that $1 - (1-\lambda)^\eta$ is a monotonically increasing function on $0 < \lambda < 1$ producing the compact domain $[0, 1]$, it is equal 0 at $\lambda = 0$ and 1 at $\lambda = 1$, and its derivative of Laplace shape, largest at $\lambda = 0$ (fastest growth) is rapidly decaying. This immediately indicates that this function saturates rapidly, and for λ', it is half way through the saturation on $[0, 1]$, by design. ■

The proof makes it explicit that the statement of the proposition on matrix convergence holds for any matrix norm. Figure 3.5 illustrates the push-forward functions underlying the Grassmann feature map and our MaxExp(F). As is clear, the Grassmann feature map binarizes the spectrum of covariance, where its cut-off $\lambda' \in \{\lambda_1, \cdots, \lambda_D\}$ decides which eigenvalues are mapped to one (*e.g.*, n leading eigenvalues), and which to zero (remaining ones). MaxExp(F) achieves the soft approximation of the Grassmann feature map for $\eta(\lambda')$.

3.2.8 Denoising Capacity of Linear Subspaces

Linear subspaces enjoy a denoising property as shown by Chen and Suter in [Chen and Suter, 2006]. Suppose a matrix $\tilde{\boldsymbol{Z}} \in \mathbb{R}^{D\times K}$ is a noise-free matrix and $\boldsymbol{A}$ is a corrupted matrix by the i.i.d Gaussian noise with the noise level σ. The additive noise is only applied to the support

Model	Backbone	*mini*-ImageNet		*tiered*-ImageNet	
		1-shot	5-shot	1-shot	5-shot
TADAM [Oreshkin et al., 2018]	ResNet-12	58.50 ± 0.30	76.70 ± 0.30	–	–
ECM [Ravichandran et al., 2019]	ResNet-12	$59.00 \pm -$	77.46 ± 0.00	$63.99 \pm -$	$81.97 \pm -$
CTM [Li et al., 2019a]	ResNet-18	62.05 ± 0.55	78.63 ± 0.06	64.78 ± 0.11	81.05 ± 0.52
LwoF [Gidaris and Komodakis, 2018b]	WRN-28-10	60.06 ± 0.14	76.39 ± 0.11	67.92 ± 0.16	83.10 ± 0.12
Prototypical Nets [Snell et al., 2017]	ResNet-12	60.37 ± 0.83	78.02 ± 0.57	65.65 ± 0.92	83.40 ± 0.65
FEAT [Ye et al., 2018]	WRN-28-10	65.10 ± 0.20	81.11 ± 0.34	70.41 ± 0.23	84.38 ± 0.16
LEO‡ [Rusu et al., 2019]	WRN-28-10	61.76 ± 0.08	77.59 ± 0.12	66.33 ± 0.05	81.44 ± 0.09
wDAE-GNN [Gidaris and Komodakis, 2019]	WRN-28-10	62.96 ± 0.15	78.85 ± 0.10	68.18 ± 0.16	83.09 ± 0.12
MetaOpt-SVM [Lee et al., 2019a]	ResNet-12	64.09 ± 0.62	80.00 ± 0.45	65.81 ± 0.74	81.75 ± 0.53
E3BM [Liu et al., 2020a]	ResNet-25	$64.30 \pm -$	$81.00 \pm -$	$70.00 \pm -$	$85.00 \pm -$
Ensemble [Kim et al., 2020]	ResNet-18	63.95 ± 0.61	81.19 ± 0.43	70.44 ± 0.32	85.43 ± 0.21
DeepEMD [Zhang et al., 2020]	ResNet-12	68.77 ± 0.29	84.13 ± 0.53	74.29 ± 0.32	86.98 ± 0.60
DSN	ResNet-12	65.38 ± 0.63	82.30 ± 0.41	69.52 ± 0.67	85.75 ± 0.48
DSN-Fast-Subspace	ResNet-12	64.67 ± 0.58	82.46 ± 0.50	69.50 ± 0.66	85.34 ± 0.52
DSN-Discriminative	ResNet-12	67.34 ± 0.62	82.51 ± 0.42	70.70 ± 0.66	86.11 ± 0.52
DSN-Grassmann-Linear	ResNet-12	69.17 ± 0.65	85.45 ± 0.75	74.09 ± 0.66	87.64 ± 0.46
DSN-Grassmann-RBF	ResNet-12	$\mathbf{69.88 \pm 0.81}$	$\mathbf{86.47 \pm 0.41}$	$\mathbf{76.10 \pm 0.67}$	$\mathbf{88.32 \pm 0.45}$
DSN-Grassmann-Fast-Subspace	ResNet-12	68.95 ± 0.62	84.81 ± 0.50	73.33 ± 0.69	87.70 ± 0.51

Table 3.1: Comparison with the state of the art. 5-way few-shot classification results with 95% confidence interval on *mini*-ImageNet and *tiered*-ImageNet datasets with various backbones for 1-shot and 5-shot protocols.

images in which the subspace approximation is performed, thus, the error is calculated between a noise-free matrix $\tilde{\boldsymbol{Z}}$ and a rank-n approximation of $\boldsymbol{A}$. The noisy matrix yields a subspace $\boldsymbol{P}' \in \mathbb{R}^{D \times n}$ from the perturbed matrix $\boldsymbol{A}$. It can be shown that the projection error for rank n is:

$$d^2(\tilde{\boldsymbol{z}}, \boldsymbol{P}') = \left\| \tilde{\boldsymbol{z}} - \boldsymbol{P}' \boldsymbol{P}'^{\top} \tilde{\boldsymbol{z}} \right\|_2^2 \simeq \sigma^2 (D - n) \sum_{i=1}^{n} \frac{h_i^2}{\lambda_i^2}, \tag{3.17}$$

where h_i is the coefficient of $\boldsymbol{h}$ corresponding to index i and $\tilde{\boldsymbol{z}} = \boldsymbol{P}\boldsymbol{h}$ by assuming $\tilde{\boldsymbol{z}} \in \text{span}(\boldsymbol{P})$, that is, $\tilde{\boldsymbol{z}}$ is constructed within the span of $\boldsymbol{P}$ according to $\boldsymbol{h}$. The desired property to have a low error is attained by having $\lambda_i \geq h_i$. Notice that the lowering of the error can be also achieved by the truncation of the smallest eigenvalues when constructing the subspace $\boldsymbol{P}'$. In contrast, the Prototypical Nets yield the following error:

$$\|\boldsymbol{\mu} - \boldsymbol{\mu}'\|_2^2 \simeq \sigma^2 D, \tag{3.18}$$

where $\boldsymbol{\mu}'$ is a prototype built from the noisy inputs. From the analysis, we can deduce that the error in Prototypical Nets [Snell et al., 2017] is greater than the error using subspace methods. This means that the error of our method can be reduced further by increasing the subspace dimension n and the singular value λ_i. We will show empirically in § 3.3.3 that the performance of Prototypical Nets plummet drastically compared to our method under the noise.

3.3 Experiments

In this section, we contrast and assess our method against state-of-the-art techniques on five challenging datasets, namely *mini*-ImageNet [Ravi and Larochelle, 2017], *tiered*-ImageNet [Ren et al., 2018a], CIFAR [Krizhevsky et al., 2009], Open MIC [Koniusz et al., 2018], and

Model	backbone	1-Shot	5-Shot
TADAM [Oreshkin et al., 2018]	ResNet-12	40.10 ± 0.40	56.10 ± 0.40
MetaOpt [Lee et al., 2019a]	ResNet-12	41.10 ± 0.60	55.50 ± 0.60
Prototypical Nets [Snell et al., 2017]	ResNet-12	41.54 ± 0.76	57.08 ± 0.76
Matching Nets [Vinyals et al., 2016]	ResNet-12	43.88 ± 0.75	57.05 ± 0.71
MTL [Sun et al., 2019b]	ResNet-12	45.10 ± 1.80	57.60 ± 0.90
DeepEMD [Zhang et al., 2020]	ResNet-12	46.60 ± 0.26	63.22 ± 0.71
DSN	ResNet-12	44.35 ± 0.62	62.82 ± 0.52
DSN-Fast-Subspace	ResNet-12	45.65 ± 0.68	62.86 ± 0.57
DSN-Discriminative	ResNet-12	45.27 ± 0.55	63.11 ± 0.60
DSN-Grassmann-Linear	ResNet-12	46.63 ± 0.59	63.83 ± 0.58
DSN-Grassmann-RBF	ResNet-12	$\mathbf{46.90 \pm 0.58}$	$\mathbf{64.46 \pm 0.69}$
DSN-Grassmann-Fast-Subspace	ResNet-12	45.60 ± 0.53	63.56 ± 0.75

Table 3.2: 5-way few-shot classification results on the FC-100 dataset using ResNet-12 with 95% confidence intervals. Methods are compared on 1-shot and 5-shot protocols.

Model	5-way 1-shot					5-way 3-shot				
	$p1 \rightarrow p2$	$p2 \rightarrow p3$	$p3 \rightarrow p4$	$p4 \rightarrow p1$	Avg	$p1 \rightarrow p2$	$p2 \rightarrow p3$	$p3 \rightarrow p4$	$p4 \rightarrow p1$	Avg
Matching Nets [Vinyals et al., 2016]	69.40	57.30	76.35	53.68	64.18	84.10	74.20	87.47	70.83	79.15
Relation Nets [Sung et al., 2018]	70.10	49.70	66.90	46.90	58.40	80.90	61.90	78.50	58.90	70.05
Prototypical Nets [Snell et al., 2017]	66.33	52.03	74.28	54.30	61.74	81.60	73.55	83.55	69.15	76.96
SoSN [Zhang and Koniusz, 2019]	78.00	60.10	75.50	57.80	67.85	87.10	72.60	85.90	72.80	79.60
DSN	74.53	58.30	76.64	60.06	67.34	86.14	74.91	87.14	73.05	80.31
DSN-Discriminative	75.87	62.13	78.25	62.11	69.59	87.93	75.78	88.42	76.59	82.18
DSN-Grassmann-Linear	78.28	**65.65**	79.98	69.51	73.35	88.47	79.66	91.18	79.87	84.79
DSN-Grassmann-RBF	**78.48**	65.13	**80.02**	**70.18**	**73.45**	**91.64**	**80.67**	**91.98**	**82.12**	**86.60**

Table 3.3: Few-shot classification results using Conv-4 on the Open MIC dataset for 5-way 1-shot and 3-shot.

CUB [Welinder et al., 2010]. We use two CNN backbones: 4-convolutional layers (Conv-4) as implemented in [Snell et al., 2017] and ResNet-12 as employed in [Lee et al., 2019a] in our experiments for standard few-shot classification. We follow a general practice to evaluate the model on the N-way K-shot protocol with 15 query images (except for Open MIC). For the perturbation analysis and evaluations on the OpenMIC dataset, the Conv-4 backbone is adopted. The reported results of our method for DSN, DSN with discriminative term, DSN using image patches (Grassmann with/without kernel methods), and DSN using the fast approximate subspace creation are provided for all datasets with the average accuracy over $2,000$ episodes. We use the ResNet-12 backbone and the image size of 84×84 pixels, as in [Lee et al., 2019a; Zhang et al., 2020], unless stated otherwise.

***mini*-ImageNet.** The *mini*-ImageNet [Ravi and Larochelle, 2017] contains 60,000 images from the ImageNet [Russakovsky et al., 2015] dataset. Images in the *mini*-ImageNet are of size 84×84, and they represent 100 classes, where 64, 16, and 20 classes are used for training, validation, and testing, respectively. Every class has 600 images following the image list from [Ravi and Larochelle, 2017].

***tiered*-ImageNet.** This dataset is also derived from ImageNet but it contains a broader set of classes compared to the *mini*-ImageNet. There are 351 classes from 20 different next-level categories for training, 97 classes from 6 different next-level categories for validation, and 160 classes from 8 different next-level categories for testing.

CIFAR-100. In our evaluations, we use the so-called FC-100 subset of the CIFAR-100 dataset. The number of samples per class is 600. FC-100 [Oreshkin et al., 2018] is a few-shot learning benchmark containing all 100 classes from CIFAR-100 [Krizhevsky et al., 2009]. The dataset is divided into 64, 20, and 20 classes for train, validation, and test, respectively.

Open MIC. This dataset [Koniusz et al., 2018] contains images from 10 museum exhibition spaces. In this dataset, there are 866 classes and 1-20 images per class. The images undergo various photometric and geometric distortions, the classes are often fine-grained in their nature, thus making few-shot learning problem challenging. The protocols and baselines we use are proposed in [Zhang and Koniusz, 2019] but exclude the easiest to classify classes, which enables going beyond 1-shot protocol, thus we rerun the SoSN [Zhang and Koniusz, 2019] baseline for fair comparisons. The dataset is divided into four subsets: *p1=(shn+hon+clv)*, *p2=(clk+gls+scl)*, *p3=(sci+nat)*, *p4=(shx+rlc)*. Protocol [Zhang and Koniusz, 2019] assumes evaluations on *p1*→*p2*, *p2*→*p3*, *p3*→*p4*, and *p4*→*p1*, where $x \rightarrow y$ denotes training on subset x and testing on subset y. Training in a subset and testing in another subset depicts a few-shot learning problem because exhibits between museums differ significantly. In experiments on this dataset, we use the Conv-4 backbone, as described in [Zhang and Koniusz, 2019; Simon et al., 2020a].

CUB. The CUB dataset [Welinder et al., 2010] contains 200 classes and 11,788 bird images in total. The dataset split used in the experiments is based on the cross-domain few-shot classification protocol from [Hilliard et al., 2018]. The splits of the dataset contain 100 base, 50 validation, and 50 novel classes. The training and evaluation procedures for cross-domain few-shot classification are adopted from [Chen et al., 2019a].

3.3.1 Few-shot Learning

We follow the general practice and evaluate our method on *mini*-ImageNet, *tiered*-ImageNet, FC-100, and Open MIC. CUB is evaluated in the cross-domain setting in the next section. For a fair comparison, we follow the experimental setup as described in [Zhang et al., 2020].

Implementation Details. In our implementations, we apply two training stages [Zhang et al., 2020; Ye et al., 2018] to obtain the models for evaluation: 1) pre-training and 2) meta-training. Firstly, the models are trained on the base classes with convolutional layers and a fully connected layer. Subsequently, we use the pre-trained model as an initialization for meta-training with episodes (N-way K-shot protocols). We use SGD for optimizing ResNet-12 in both pre-training and meta-training stages. During meta-training, we remove the FC layer and train our models using the Discriminative Subspace Networks to maximize the distance between class-wise subspaces (maximizing the inter-class distance). After the training stage, we initialize an FC layer randomly for each episode. Then, this FC layer and the last convolutional layer are updated with 150 iterations using SGD. To create image patches, we apply random crops and sample 25 patches per image following the strategy in [Zhang et al., 2020]. For the RBF kernel, γ is set to 2.0, 2.0, 1.0, and 2.0 on *mini*-ImageNet, *tiered*-ImageNet, FC-100, and CUB→*mini*-ImageNet

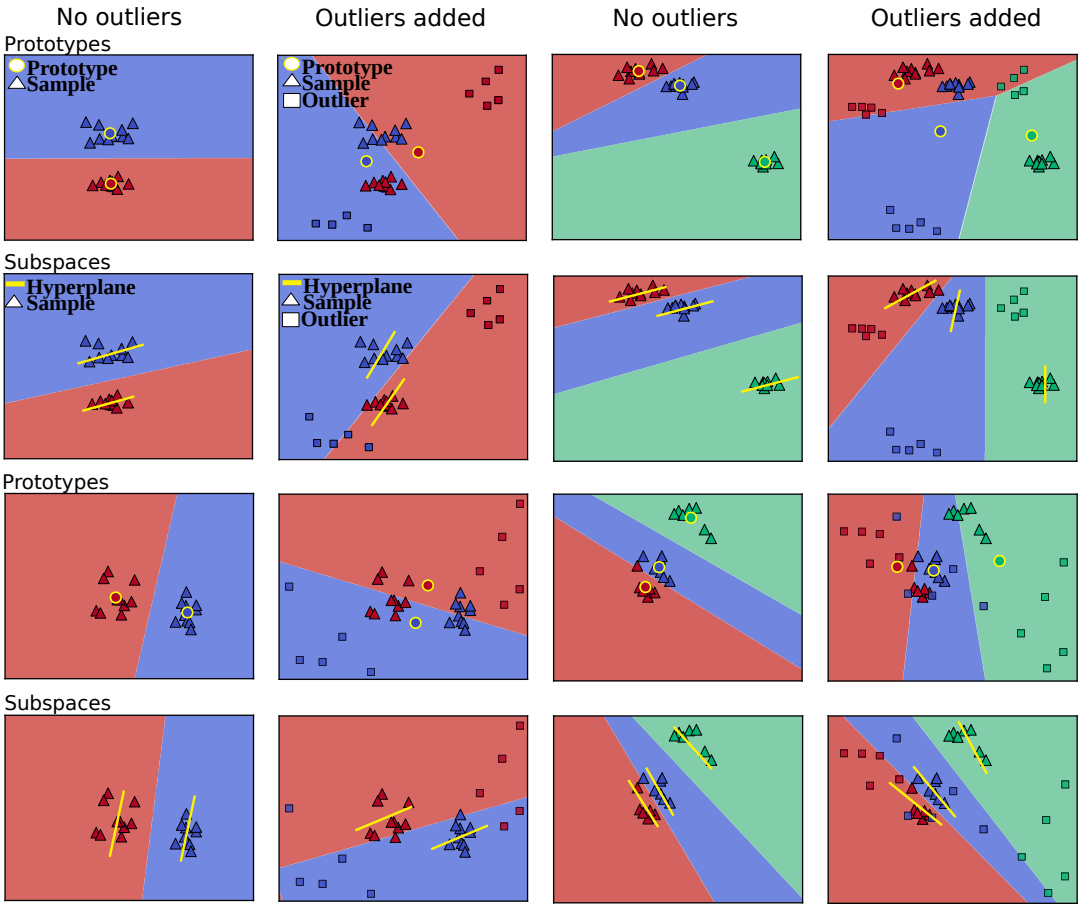


Figure 3.6: The impact of outliers on prototypes and subspaces. The odd rows show the decision boundaries obtained via prototypes (with and without outliers) for two- and three-class problems. The even rows depict how subspaces behave for the same problems. In general, subspaces show a better resilience to perturbations and attain higher discriminative power in comparison to prototypes.

after cross-validation, respectively. For the fast subspace creation, we set $\eta = 20$ for all experiments. Moreover, $\lambda = 0.03$ in for all experiments after cross-validation.

By design, our method for the distance from a latent query point to a subspace needs more than one sample to identify the span of a subspace. Thus, for the 1-shot case, we generate an additional sample by data augmentation through flipping support images.

Results. Below, we provide our results based on the ResNet-12 backbone for a comprehensive comparison. Note that different backbones can affect the performance of few-shot learning. For the *mini*-ImageNet, Table 3.1 shows that our method outperforms state-of-the-art methods for 5-way 1-shot and 5-way 5-shot protocols. We gain 1.11% and 2.34% improvement on these protocols compared to DeepEMD on *mini*-ImageNet and 1.81% and 1.34% improvement on *tiered*-ImageNet. Our method consistently outperforms the other methods on the *mini*-ImageNet, *tiered*-ImageNet and FC-100 datasets (see Tables 3.1 and 3.2).

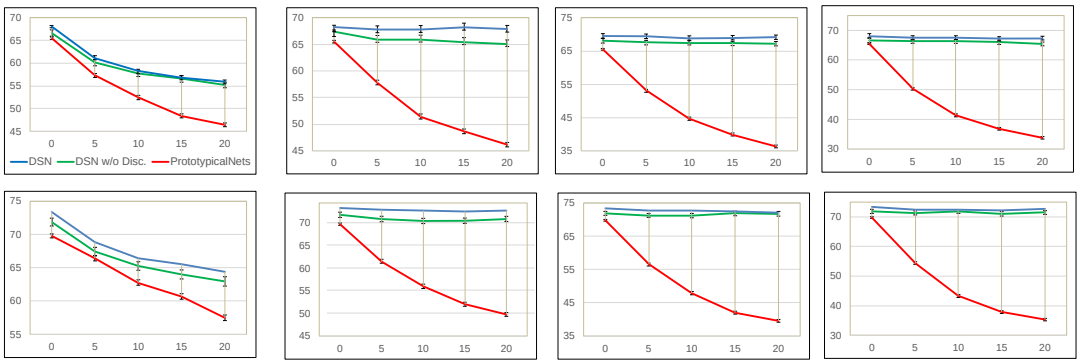


Figure 3.7: Experiments in the presence of outliers and additive noise on *mini*-ImageNet for 5-way 5-shot and 10-shot protocols using the Conv-4 backbone. The results of DSN, DSN without discriminative term, and Prototypical Nets are given (see the legend). The *first column* shows the impact of introducing outliers among support samples (the classes of outliers are disjoint with the support classes of samples). The *second, third and fourth* columns show the impact of introducing noisy samples generated randomly according to the Gaussian distribution with random means and variance of $\sigma = \{0.15, 0.3, 0.4\}$, respectively. The performance is measured w.r.t. the increasing number of outliers and noisy samples (x-axis).

3.3.2 Cross-Domain Experiments

In what follows, we apply the cross-domain few-shot learning setup in [Chen et al., 2019a], as this is one of established cross-domain benchmarks with numerous pipelines reporting their results on it. The datasets used for training and testing the model are the *mini*-ImageNet and CUB datasets, respectively. We compare our method with DeepEMD [Zhang et al., 2020], Prototypical Nets [Snell et al., 2017], Matching Nets [Vinyals et al., 2016], K-nearest neighbors, the cosine classifier, and the linear classifier. Table 3.4 shows that our method leads by 3.61% and 1.25% on 5-way 1-shot and 5-way 5-shot protocols, respectively. We also use the Open MIC dataset, another pioneering cross-domain few-shot learning benchmark. Of course, there exist more datasets proposed recently for the cross-domain task alone, which are beyond the scope of this paper. On the Open MIC dataset (see Table 3.3), a similar trend can be observed. Our methods outperform state-of-the-art embedding methods for few-shot learning (*i.e.*, , Matching Nets [Vinyals et al., 2016], Prototypical Nets [Snell et al., 2017], and Second-order Similarity Network (SoSN) [Zhang and Koniusz, 2019]). The results show that our subspace representation is robust to various photometric and geometric distortions posed by the Open MIC dataset, and it can model well the diversity of objects from each domain *e.g.*, mechanical installations, glassware, pottery, relic artefacts, wooden sculptures, paintings, skeletons of animals, etc. Our model generalizes to different subsets of exhibits on Open MIC with over 5% gain on average compared to other methods.

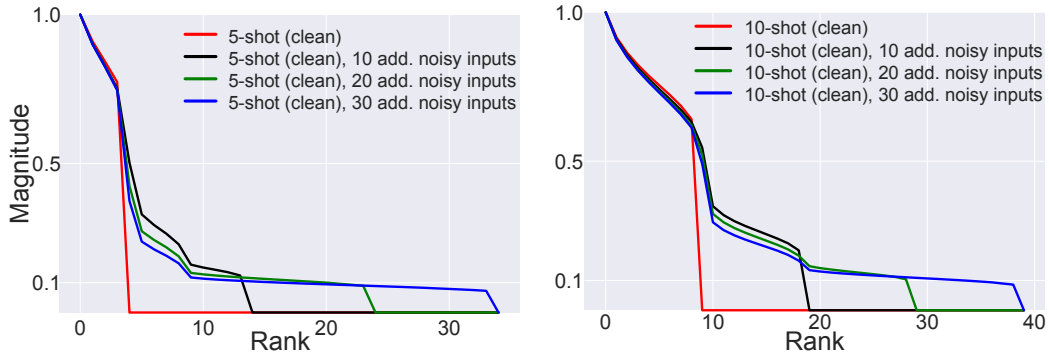


Figure 3.8: The analysis of singular values of class-wise feature matrices. Given the 5-shot protocol, we replicate 5 images per class $2\times$, $4\times$ and $6\times$, and add the noise to these replicated images. We ℓ_2-norm normalize all support feature vectors, we apply SVD to each matrix (5 matrices for 5 classes), and we average the spectra. The results for (a) the 5-way 5-shot and (b) the 5-way 10-shot protocols show that the discriminative information is carried in the leading $K-1$ eigenvalues, as they carry significantly more energy compared with remaining singular eigenvalues, where the added noise is concentrated. Thus, choosing the right subspace size (parameter n) helps filter out the noise.

3.3.3 Robustness Study

Subspaces *vs*. Prototypes on the Toy Data. Below, we analyze the robustness of our approach on toy data. We visualize how our subspace-based method compares to the prototype-based baseline under existence of the noise and outliers. Fig. 3.6 shows that the subspace classifier is more robust to outliers than the prototype-based baseline [Snell et al., 2017]. Outliers have a dramatic impact on prototypes resulting in their large drift, and the misclassification as a result. However, our subspace-based approach is less perturbed by the noise and outliers while exhibiting a better discriminative power due to its ability to capture second-order statistics.

Robustness to Perturbations. One may argue whether noise poses problems in few-shot learning. The presence of noise patterns might not be obvious when collecting the data. Thus, the data cannot be guaranteed to be free from the noise. We observed in our experiments that the performance for standard methods degrades significantly with a small degree of perturbations added to signal, as depicted in Fig. 3.7. However, our subspace-based model handles such a noise well.

Analysis of Singular Values on a Perturbed Set. Below, we investigate the impact of adding the noise on the singular values from a within-class set of clean and noisy inputs. In this analysis, we use clean samples in episodes, replicate them $2\times$, $4\times$ and $6\times$, and add to them the noise from the Normal distribution ($\sigma = 0.3$). Afterwards, a set of samples is decomposed to obtain singular value matrices, one per class, whose spectra are averaged. Fig. 3.8 shows (*mini*-ImageNet) the singular values from a set of clean images as we gradually add more copies contaminated with the noise. As the first $n = K-1$ singular values appear mostly unchanged between clean and noisy cases, the n leading singular values and their corresponding vectors span the discriminative subspace. The above observation explains the robustness of our dynamic classifier.

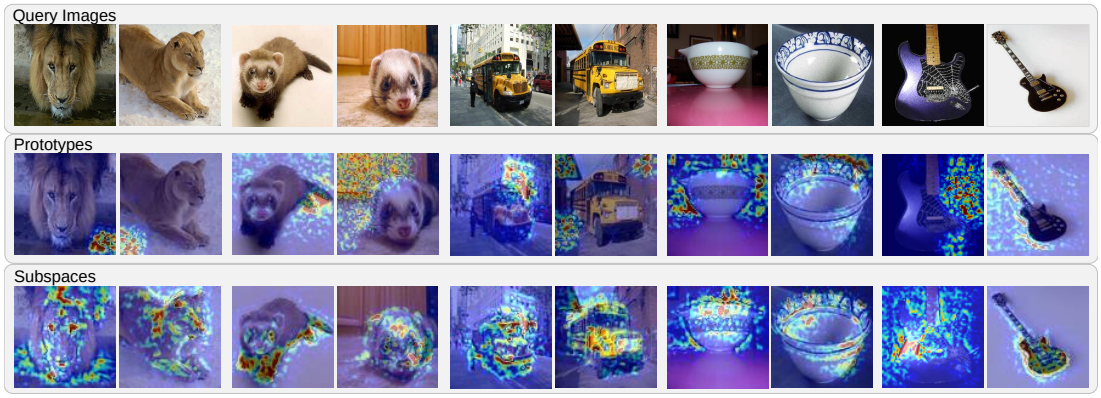


Figure 3.9: Visualization using GradCam [Selvaraju et al., 2017] (a gradient-based method) to show the attention of the query images given the support images. The distance for the objective function is the euclidean distance between a query and either a prototype (middle) or a projection onto a subspace (bottom).

3.3.4 Ablation Studies

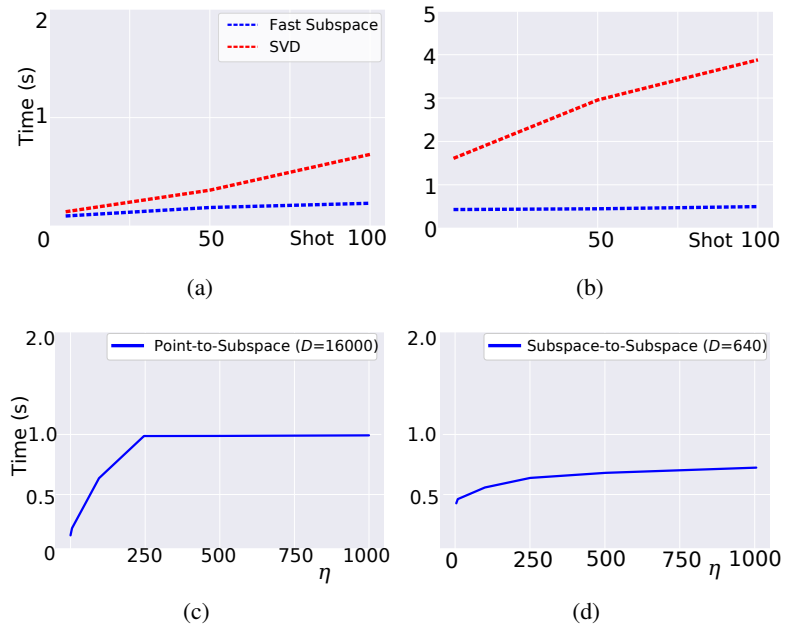
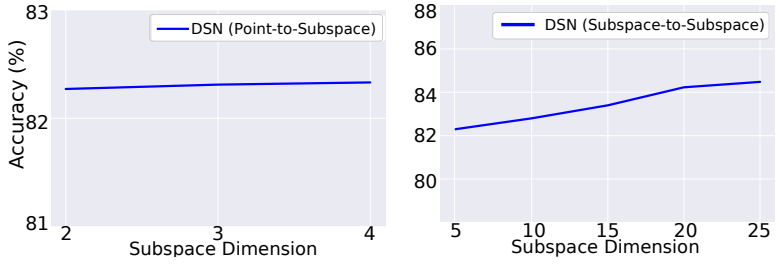


Figure 3.10: (Top) Time comparisons of SVD *vs*. fast subspace creation w.r.t. the shot number. (Bottom) The running time by varying η for the fast subspace creation. (a), (c) Computing the distance between a query point to its projection on a subspace. (b), (d) Computing the distance between two subspaces (Grassmann). The result is reported for the 5-way 5-shot protocol on *mini*-ImageNet. Note that (a), (c) use $D = 16000$ whereas (b), (d) use $D = 640$.

Model	backbone	1-Shot	5-Shot
Cosine Classifier [Chen et al., 2019a]	ResNet-12	44.17 ± 0.78	69.01 ± 0.74
Linear Classifier [Chen et al., 2019a]	ResNet-12	50.37 ± 0.79	73.30 ± 0.69
Matching Nets [Vinyals et al., 2016]	ResNet-12	51.65 ± 0.84	69.14 ± 0.72
Prototypical Nets [Snell et al., 2017]	ResNet-12	50.01 ± 0.82	72.02 ± 0.67
KNN [Li et al., 2019b]	ResNet-12	50.84 ± 0.81	71.25 ± 0.69
DeepEMD [Zhang et al., 2020]	ResNet-12	54.24 ± 0.86	78.86 ± 0.65
DSN	ResNet-12	55.27 ± 0.65	76.76 ± 0.57
DSN-Fast-Subspace	ResNet-12	56.48 ± 0.66	77.41 ± 0.58
DSN-Discriminative	ResNet-12	56.99 ± 0.71	77.71 ± 0.57
DSN-Grassmann- Linear	ResNet-12	56.22 ± 0.39	78.38 ± 0.66
DSN-Grassmann-RBF	ResNet-12	$\mathbf{57.85 \pm 0.65}$	$\mathbf{80.11 \pm 0.51}$
DSN-Grassmann-Fast-Subspace	ResNet-12	57.65 ± 0.65	78.28 ± 0.49

Table 3.4: 5-way cross-domain few-shot classification results trained on the *mini*-ImageNet dataset and evaluated on the CUB dataset using ResNet-12 with 95% confidence intervals.



(a) (b)

Figure 3.11: The 5-way 5-shot performance on *mini*-ImageNet using DSN point-to-subspace and subspace-to-subspace approaches w.r.t. the dimensionality of subspaces denoted by n.

Visualization. Below we use GradCam [Selvaraju et al., 2017] to visualize the heatmap (for subspace- and prototype-based approaches) on input images to understand which parts of images participate in matching. In this setting, we use a pre-trained ResNet-12 on *mini*-ImageNet with base classes, and we apply the visualization on the unseen classes. In order to form prototypes and subspaces, we select 5 samples per class as a support set and show the heatmap of query images. Fig. 3.9 visualizes the heatmap using the distance between prototypes and the projection onto the subspace as a loss function. As shown in Fig. 3.9, the heatmap using our subspace method agrees with those objects in the support sets. In contrast, prototypes based on average pooling result in responses that do not cover well the objects or interest or even fire on backgrounds.

Subspace Dimensionality. In comparison to other models such as Matching Nets, Prototypical Nets, and Relation Nets, our DSN comes with an additional hyper-parameter, the dimensionality of the subspaces denoted by n. As a rule of thumb, we recommend to use $n = K - 1$ for DSN to train and test our model and $n = H - 1$ for DSN with image patches. In Fig. 3.11, DSN exhibits a large degree of robustness to n, for our point-subspace method. We observe that

the choice of n from 2 to $K-1$ does not affect the performance significantly ($\pm$ 0.5%) on *mini*-ImageNet using ResNet-12 backbone. However, the dimensionality of DSN using image patches affects the performance by 2%, which suggests that the use of patches results in a much more informative spectrum of covariances.

Computational Complexity. In our work, we can create a subspace from a set of K samples and a set of H image patches. We use k' to generalize the complexity analysis to both cases. The computational complexity of our DSN approach is $\mathcal{O}(\min(ND^2k', NDk'^2))$, where N and D are the way number and the feature dimensionality, respectively. Compared to the complexity of the Prototypical Nets *i.e.*, $\mathcal{O}(NDk')$, our method is somewhat slower due to the use of SVD. However, fast approximate SVD algorithms can be used [Menon and Elkan, 2011]. A fast approximate Grassmann feature map can be obtained via the spectral MaxExp(F) algorithm [Koniusz and Zhang, 2020b], which relies only on a small number of matrix-matrix multiplications, sub-linear w.r.t. its parameter η, which rivals for example the Newton Schulz iterations used for computing the matrix square root [Lin and Maji, 2017]. To this end, we employed a fast subspace creation, as described in § 3.2.7. In the case of the subspace method for few-shot classification (point-to-subspace), the complexity of using matrix-matrix multiplication reduces the time using SVD from $\mathcal{O}(\min(ND^2k', NDk'^2))$ to $\mathcal{O}(NDk'\eta)$. For efficient matrix-matrix multiplications parallelized on the GPU, the subspace-to-subspace variant of our approach also enjoys a very attractive complexity $\mathcal{O}(NDk' \log \eta)$. In our experiment, we show that Fig 3.10 shows that the fast subspace creation is indeed reducing the computational complexity while keeping the runtime low for various shot numbers. In contrast, the computational complexity of SVD grows exponentially as the shot number increases.

Limitations. The subspace method needs to have more than one datapoint to create the projection matrix, otherwise the approach reduces to the prototypical solution. Therefore, we use a simple data augmentation strategy *e.g.*, we apply the image flip to create a secondary sample for 1-shot case. We have observed empirically that this strategy does not give a huge improvement boost to the performance of other methods such as Prototypical Nets [Snell et al., 2017] but it helps us to create a valid subspace in such a case. Moreover, we also propose how to use patches and create a set of features for constructing a subspace. We treat patches as an orderless set of local descriptors but more advanced approaches could in addition consider their spatial locations relative to each other and the location in the global scene.

3.4 Chapter Conclusion

A novel few-shot learning approach based on point-to-subspace and subspace-to-subspace modeling. We have shown that the representations learned via DSN are expressive across numerous few-shot problems and benchmarks. The models are trained on base classes and evaluated on the test set whose class concepts were excluded from the training classes. The subspace model outperforms the existing models by a large margin due to its ability of representing a few of datapoints on a subspace, enjoying second-order statistics and denoising capacity.

In DSN, each class classifier is represented by the subspace formed by all its samples, meaning that each class is modeled by the span of its training datapoints. In addition, our

method applies the distance calculation for a set of subspaces on Grassmann manifolds yielding performance improvement on various FSL benchmarks. We have shown theoretically and empirically why DSN is robust to noise. Finally, we have included additional variants of our approach based on subspaces combined with RKHS kernels and fast subspace approximations, each showing different advantages depending on the benchmark regime at hand.

Chapter 4

Multi-Label Propagation for Multi-Label Few-Shot Learning

In this chapter, we introduce the problem of multi-label few-shot learning. The problem is addressed using a label-propagation method. The data is formed into a graph, where the nodes are composed of the support samples. Besides, we also propose a label counting method that can estimate the number of objects in an image. Considering the label propagation and the label counting modules, our experiments show that we can improve the performance upon baselines.

4.1 Introduction

In machine learning, the episodic learning [Santoro et al., 2016; Vinyals et al., 2016] has been shown to be beneficial in learning novel concepts from limited data. Despite the success, the conventional form of episodic learning comes with a somehow limiting assumption as the learning is formulated purely for single-label problems. In other words, though multi-class, each data sample can only belong to one of possible classes. In the case of images, this means that each image should just encapsulate one visual concept (*i.e.*, object). This clearly limits the application of the developed techniques in many places (*e.g.*, fashion recognition [Inoue et al., 2017], multimedia content analysis [Qi et al., 2007; Pachet and Roy, 2009], bioinformatics [Barutcuoglu et al., 2006; Cesa-Bianchi et al., 2012], and drug discovery [Heider et al., 2013] to name a few). This work tries to bridge the gap by introducing techniques to address Multi-Label Few Shot Learning (ML-FSL).

To get a feeling about the difficulty of ML-FSL, we recall that a profound idea in single-label FSL is to make use of relational comparisons. In essence, one learns to measure the similarity between pairs of samples, where a pair constitutes of the query and an example from the supporting set. While comparing samples in the case of single-labels is well-behaved, extension to the multi-label regime is not an easy ride (see Fig. 4.1 for a conceptual diagram). This is because, by merely inspecting the similarity between pair of samples, one cannot deduce the class labels of the query.

In this chapter, we extend and introduce novel methods to tackle the problem of ML-FSL. In particular, we investigate three methods with different characteristics:

1. Multi-Label Prototypical Networks. Inspired by the work of [Snell et al., 2017], we make use of the notion of class prototypes to tackle the problem of ML-FSL.

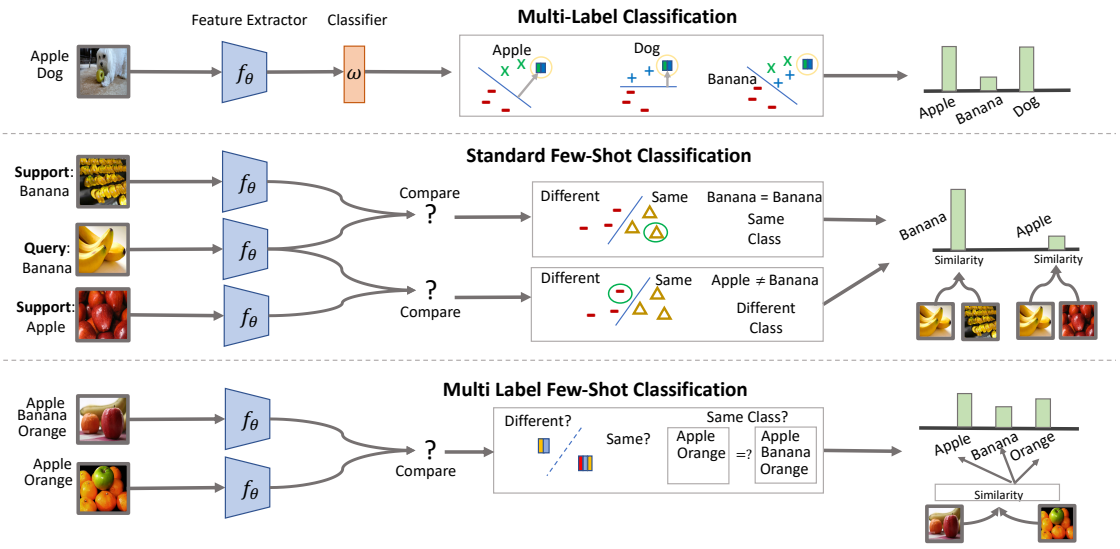


Figure 4.1: Comparison between multi-label classification, single-label few-shot learning and multi-label few shot learning. Multi-label few-shot problem. **Top panel.** In classical multi-label classification, one designs a fixed classifier from all seen classes. In a well-established practice, one breaks down the problem into identifying a set of binary classifiers where each classifier is responsible for identifying one specific class in a given input. **Middle panel.** A profound idea in addressing single-label FSL is to design a model to perform relational comparisons. Here, the model receives pairs of images (the query and an image from the support set) and predicts whether they are similar or not. We note that while in multi-label problems, the embedding of the query image is fixed, in relational inference, the embedding is dependent on the constructed pairs. **Bottom panel.** Extension of the relational inference to multi-label FSL regime is not trivial. Our work provides various solutions to address this challenging, yet extremely important and practical problem.

2. Multi-Label Relation Networks. Building upon the work of [Sung et al., 2018], we introduce multi-label relational networks with binary relevance.

3. Label Propagation Networks. We propose a label propagation algorithm to address the problem of ML-FSL. Label propagation models the data in the form of graphs. This model learns the correlations among samples by weighting mechanism according to their similarities.

To the best of our knowledge, only the work of [Alfassy et al., 2019] tackles the problem of ML-FSL. Our work goes beyond this work in various ways. This includes, extending and introducing new algorithms for ML-FSL, developing a comprehensive evaluation framework with three challenging datasets, namely MS-COCO [Lin et al., 2014], iMaterialist [Guo et al., 2019], and Open MIC [Koniusz et al., 2018] and proposing a neural model to perform label count, a module that empirically seems to be boosting the accuracy by a tangible margin.

To summarize, our contributions in this paper are:

i. By departing from single-label FSL problems, we generalize and introduce various techniques to address the ML-FSL problem.

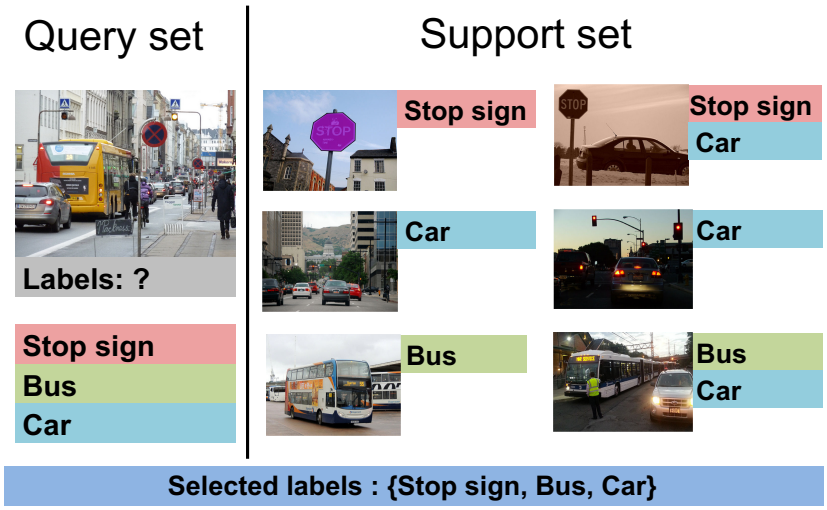


Figure 4.2: An episode consists of a query set and a support set. The support set contains examples from selected classes of a given task. The query set covers the same set of labels presented in the support set.

ii. A multi-label propagation method is proposed to weigh the features of each sample based on importance.

iii. We propose a neural label count module (NLC) to estimate the number of labels in an input and thoroughly asses its efficiency in conjunction with the proposed ML-FSL solutions.

4.2 Problem Definition

Our goal is to construct meta-learning tasks such that the models can gain experience or learn from similar tasks. Inspired by [Vinyals et al., 2016], we propose the concept of multi-label few-shot classification via learning from *episodes*. Below, we explain our setting, notations, training, and testing strategies for ML-FSL.

Throughout this paper, we use bold lower-case letters (*e.g.*, $\boldsymbol{x}$) to denote column vectors and bold upper-case letters (*e.g.*, $\boldsymbol{X}$) to denote matrices. A network is denoted as $f_\Theta : \mathbb{R}^m \rightarrow \mathbb{R}^n$ which maps an input into some feature space. The set of labels per *episode* is represented by $\mathcal{C}$. Let $\mathcal{X} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_{N_x}, \boldsymbol{y}_{N_x})\}$ with $\boldsymbol{x}_i \in \mathbb{R}^n; \boldsymbol{y}_i \in \{0,1\}^{|\mathcal{C}|}$ and $\mathcal{Q} = \{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_{N_q}\}$ with $\boldsymbol{q}_i \in \mathbb{R}^n$ denote the support set and query set, respectively.

Episode. To form an episode, we need to have a task distribution T over possible label sets, then, a set of labels $\mathcal{C}$ is sampled from T. There are two sets shaping an *episode*: a support set $\mathcal{X}$ and a query set $\mathcal{Q}$. To address the multi-label classification, $\boldsymbol{y}_i \subseteq \mathcal{C}$ denotes a set of multiple labels assigned to $\boldsymbol{x}_i$. The network is assigned an *episode* in order to output an appropriate set of labels $\boldsymbol{y}_i$ for each $\boldsymbol{q}_i$. Note that, an episode composition including selected classes and examples may differ from one episode at a given timestep to another. This *episode* structure exploits meta-knowledge such that a model that has the meta-learning capability will learn to

match representations regardless of the actual semantic meaning of the labels, thus learning a relation between data points. As a result, the models can generalize and predict the appropriate labels given only a few data points during training. The challenges such a technique has to address are twofold: (i) the number of data points is low and (ii) the size of the label set varies from image to image thus making the prediction task harder.

N-way K-shot. In single-label few-shot learning, the term N-way K-shot is used to describe the number of classes (way) in an episode and the number of examples (shot) in each class. Following this notation, we also sample N-way and K-shot from each label to form an *episode*. For example, suppose there are three sampled classes ($N = 3$), then we obtain five examples ($K = 5$) per class to compose an *episode*, so the support set contains 15 images in total. However, the support set composition for multi-label few-shot learning differs from the single-label setting in the sense that the distribution over samples per class in an episode is not uniform. Each data point $\boldsymbol{x}_i$ in a support set can contain more than one label from the set of labels $\mathcal{C}$, thus, it is not guaranteed that there are exactly $K \times N$ samples constituting the support set of an episode.

Training Stage. In majority of cases, to perform multi-label classification, pre-trained CNNs are used [Wang et al., 2016, 2017; Zhu et al., 2017]. However, in this paper, we want to focus on the learning techniques for which the network must exhibit the capacity to improve its performance by learning from previous tasks. As a result, we train the networks from scratch by random initialization. The models are updated based on the training objective as follows:

$$\Theta^* = \arg\max_{\Theta} E_{\mathcal{C}\sim T}\left[E_{\mathcal{X}\sim\mathcal{C},\mathcal{Q}\sim\mathcal{C}}\left[\sum_{\boldsymbol{q}\in\mathcal{Q}} \log P_{\Theta}(\boldsymbol{y}|\boldsymbol{q},\mathcal{X})\right]\right], \quad (4.1)$$

where Θ denote the model parameters that we want to learn. The model is trained over many episodes to minimize the prediction error over the query set $\mathcal{Q}$.

Testing Stage. In the testing stage, the model performs classification via assigning each data point to unseen labels from a distribution T'. Following an *episode* composition in the training stage, there are N labels and at least K examples per label. Thus, one can think of such classification strategy as transfer learning from previously learnt tasks to the new set of tasks.

4.3 Multi-Label Propagation Networks

In this section, we develop an approach for ML-FSL via label propagation. Our goal is to predict the label set of a query via a neural network that discovers correlations between the support and query images. To this end, we propose the Multi-Label Propagation Network (MLPN) and make use of episodic learning to train it.

An MLPN consists of three main blocks; **i)** the feature extractor network f_{Θ} (*e.g.*, CNN), **ii)** a multi label-propagation cell and **iii)** an MLP classifier (see Fig. 4.5 for a conceptual diagram).

The feature extractor, as the name implies, is a non-linear mapping from the input space to an n-dimensional embedding space. The multi-label propagation cell receives the embedded samples from the feature extractor and generates *concept scores*. For a support set representing C classes, a concept score is a C-dimensional vector encapsulating the belief of the network in

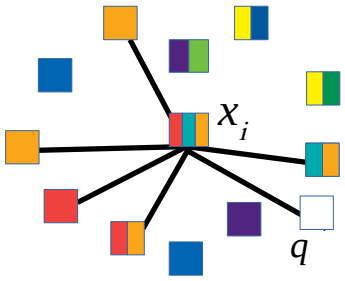


Figure 4.3: An illustration of neighborhoods from x_i. Every selected neighbor is connected by a line. The color-codes represent different labels.

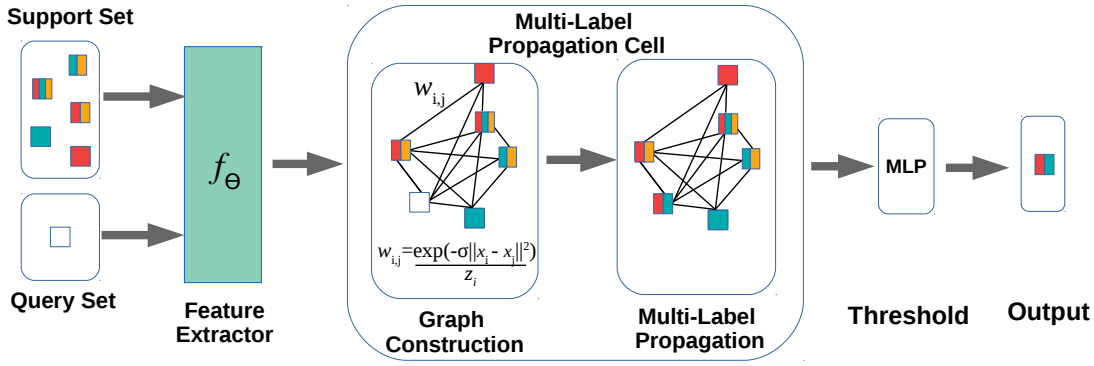


Figure 4.4: The architecture of multi-label propagation networks (MLPN). There are three modules in the proposed approach: 1). feature extractor, 2). multi-label propagation cell, and 3). an automatic threshold using an MLP.

the existence of any of the classes in the input. The MLP classifier further process the concept score and generates the final output of the network, a C-dimensional binary vector showing which classes are presented in the input.

Focusing on the multi-label propagation cell, we recall that In ML-FSL, the multi-label propagation cell sees a support $\mathcal{S} = \{\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_{N_s}\}$ with $\boldsymbol{s}_i \in \mathbb{R}^n$, the associated label-sets $\mathcal{Y} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_{N_s}\}$ with $\boldsymbol{y}_i \in \mathbb{R}^C$ and a query set $\mathcal{Q} = \{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_{N_q}\}; \boldsymbol{q}_i \in \mathbb{R}^n$ and produces the concept score matrix $\mathbb{R}^{C \times N_q} \ni \boldsymbol{\Psi}_Q = [\boldsymbol{\psi}_1, \boldsymbol{\psi}_2, \cdots, \boldsymbol{\psi}_{N_q}]$. The multi-label propagation cell ensures that $\boldsymbol{\psi}_i(j) \geq 0; \forall i, j$ and $\mathbf{1}_C^\top \boldsymbol{\psi}_i = 1$.

To predict the concept scores, we make use of the property of ***smoothness***, meaning that similar instances should have similar concept scores. More specifically, if a query sample $\boldsymbol{q}_j$ is similar to a support instance $\boldsymbol{s}_i$, then $\boldsymbol{\psi}_j \approx \nu(\boldsymbol{y}_i)$ where $\nu : \mathbb{R}^C \to \mathbb{R}^C$ is the ℓ_1 normalization operator defined as $\nu(\boldsymbol{y}) = \boldsymbol{y}/\|\boldsymbol{y}\|_1$. We define $\mathbb{R}^{n \times (N_s + N_q)} \ni \boldsymbol{X} = [\mathcal{S}, \mathcal{Q}] = [\boldsymbol{s}_1, \boldsymbol{s}_2, \cdots, \boldsymbol{s}_{N_s}, \boldsymbol{q}_1, \boldsymbol{q}_2, \cdots, \boldsymbol{q}_{N_q}]$. We denote the columns of $\boldsymbol{X}$ with $\boldsymbol{x}_i; 1 \leq i \leq N_s + N_q$. Similarly, we define $\mathbb{R}^{C \times (N_s + N_q)} \ni \boldsymbol{\Phi} = [\nu(\boldsymbol{y}_1), \nu(\boldsymbol{y}_2), \cdots, \nu(\boldsymbol{y}_{N_s}), \boldsymbol{\psi}_1, \boldsymbol{\psi}_2, \cdots, \boldsymbol{\psi}_{N_q}]$

To achieve our goal, we start by building a neighborhood graph over the support and the query sets using $\boldsymbol{X}$. Each vertex in this graph corresponds to a sample in $\mathcal{X} \cup \mathcal{Q}$ and the edge between two vertices is defined as

$$w_{i,j} = \begin{cases} \frac{1}{Z_i} \exp\left(-\sigma \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2\right), & j \in \mathcal{N}_i \\ 0, & \text{otherwise.} \end{cases} \tag{4.2}$$

Here, $\mathcal{N}_i$ denotes the neighborhood of sample i and

$$Z_i = \sum\nolimits_{j \in \mathcal{N}_i} \exp\left(-\sigma \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2\right) .$$

Remark 1. *In constructing the neighborhood graph, if a sample is drawn from the support set for which a label set exists, then the label set is used to identify its neighbors. For instance, if $\boldsymbol{x}_i$ is labelled with "car" and "bus", then every other sample labelled as either car or bus is considered in $\mathcal{N}_i$. However, if the sample is from the query set, then all the samples in the support set are considered as its neighborhood. Fig. 4.3 shows that a datapoint forms connections to others in the graph.*

Remark 2. *In our experiments and following general practice, we set the value of σ as*

$$\sigma = \underset{j \in \mathcal{N}_i}{\text{median}} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 \tag{4.3}$$

With the above, we formulate the multi-label propagation on the graph via the loss function;

$$\mathcal{L} := \sum_{i=N_s+1}^{N_s+N_q} \sum_{j=1}^{|C|} \left(\phi_i(j) - \sum_r w(i,r)\phi_r(j)\right)^2 , \tag{4.4}$$

where ϕ_j is the j-th column of $\boldsymbol{\Phi}$. Our goal is to obtain $\phi_i; N_s < i \leq N_s + N_q$ and hence $\boldsymbol{\Psi}_Q$ by minimizing Eq. (4.4) with the following additional constraints

$$\phi_i(j) \geq 0, \qquad \sum_j \phi_i(j) = 1 . \tag{4.5}$$

In essence, the loss defined in Eq. (4.4) seeks the minimum differences among concept scores of neighbouring samples. In the optimal case, the concept scores of any pair of instances that are closely connected in the graph will be similar.

An intriguing property of Eq. (4.4) is that it has a closed-form solution. Define the graph Laplacian matrix $\mathbb{R}^{(N_x+N_q)\times(N_x+N_q)} \ni \boldsymbol{L} = \mathbf{I} - \boldsymbol{W}$ where $\boldsymbol{W}$ is the adjacency matrix with element (i,j) being $w_{i,j}$ according to Eq. (4.2). The Laplacian matrix $\boldsymbol{L}$ has the following form;

$$\boldsymbol{L} = \begin{bmatrix} \boldsymbol{L}_{\mathcal{X}\mathcal{X}} & \boldsymbol{L}_{\mathcal{X}\mathcal{Q}} \\ \boldsymbol{L}_{\mathcal{Q}\mathcal{X}} & \boldsymbol{L}_{\mathcal{Q}\mathcal{Q}} \end{bmatrix} ,$$

where $\boldsymbol{L}_{\mathcal{X}\mathcal{X}} \in \mathbb{R}^{N_x \times N_x}$, $\mathbb{R}^{N_x \times N_q} \ni \boldsymbol{L}_{\mathcal{Q}\mathcal{X}}$ and $\boldsymbol{L}_{\mathcal{Q}\mathcal{Q}} \in \mathbb{R}^{N_q \times N_q}$. It can be shown that the

minimum of Eq. (4.4) is attained with

$$\boldsymbol{\Psi}_{\mathcal{Q}}^{*} = -\left((\boldsymbol{L}_{\mathcal{Q}\mathcal{Q}})^{-1}\boldsymbol{L}_{\mathcal{Q}\mathcal{X}}\boldsymbol{\Psi}_{\mathcal{X}}^{\top}\right)^{\top}. \tag{4.6}$$

Here, $\boldsymbol{\Psi}_{\mathcal{X}} = [\nu(\boldsymbol{y}_1), \nu(\boldsymbol{y}_2), \cdots, \nu(\boldsymbol{y}_{N_s})]$.

Consider the loss function:

$$\mathcal{L} \coloneqq \sum_{i=N_s+1}^{N_s+N_q}\sum_{j=1}^{|C|}\left(\phi_i(j) - \sum_{r} w(i,r)\phi_r(j)\right)^2. \tag{4.7}$$

Eq. (4.7) and its constrains are convex such that a global minimum exists. Eq. (4.7) can be transformed into a matrix form. We define $\mathbb{R}^{C\times(N_s+N_q)} \ni \boldsymbol{\Phi} = [\boldsymbol{\Psi}_{\mathcal{X}}, \boldsymbol{\Psi}_{\mathcal{Q}}]$, where $\boldsymbol{\Psi}_{\mathcal{X}} = [\nu(\boldsymbol{y}_1), \nu(\boldsymbol{y}_2), \cdots, \nu(\boldsymbol{y}_{N_s})]$ and $\boldsymbol{\Psi}_{\mathcal{Q}} = [\boldsymbol{\psi}_1, \boldsymbol{\psi}_2, \cdots, \boldsymbol{\psi}_{N_q}]$. The adjacency matrix is denoted as $\boldsymbol{W} \in \mathbb{R}^{(N_s+N_q)\times(N_s+N_q)}$. The matrix form of Eq. (4.7) with Lagrange function is formulated as follows:

$$\mathcal{L} = \frac{1}{2}\left\|\boldsymbol{\Phi}^{\top} - \boldsymbol{W}\boldsymbol{\Phi}^{\top}\right\|^2 - \boldsymbol{\alpha}^{\top}\left(\boldsymbol{\Phi}^{\top}\mathbf{1} - \mathbf{1}\right). \tag{4.8}$$

In Eq. (4.8), we can modify the first term by $\mathbb{R}^{(N_x+N_q)\times(N_x+N_q)} \ni \boldsymbol{L} = \mathbf{I} - \boldsymbol{W}$. Then, the loss function has a derivative w.r.t. $\boldsymbol{\Phi}^{\top}$ expressed as:

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{\Phi}^{\top}} = \boldsymbol{L}^{\top}\boldsymbol{L}\boldsymbol{\Phi}^{\top} - \boldsymbol{\alpha}\mathbf{1}^{\top} = \mathbf{0}. \tag{4.9}$$

Note that, $\boldsymbol{\Phi}^{\top}\mathbf{1} = \mathbf{1}$ and $\boldsymbol{L}\mathbf{1} = \mathbf{0}$, thus, $\boldsymbol{\alpha}$ can be calculated as follows:

$$\boldsymbol{L}^{\top}\boldsymbol{L}\boldsymbol{\Phi}^{\top}\mathbf{1} = \boldsymbol{\alpha}\mathbf{1}^{\top}\mathbf{1}. \tag{4.10}$$

As a result, we can imply that $\boldsymbol{\alpha} = \mathbf{0}$. Moreover, we recall that $\boldsymbol{L}$ has the following form:

$$\boldsymbol{L} = \begin{bmatrix}\boldsymbol{L}_{\mathcal{X}\mathcal{X}} & \boldsymbol{L}_{\mathcal{X}\mathcal{Q}}\\ \boldsymbol{L}_{\mathcal{Q}\mathcal{X}} & \boldsymbol{L}_{\mathcal{Q}\mathcal{Q}}\end{bmatrix}.$$

Thus, Eq. (4.9) yields the following:

$$\begin{bmatrix}\boldsymbol{L}_{\mathcal{X}\mathcal{X}} & \boldsymbol{L}_{\mathcal{X}\mathcal{Q}}\\ \boldsymbol{L}_{\mathcal{Q}\mathcal{X}} & \boldsymbol{L}_{\mathcal{Q}\mathcal{Q}}\end{bmatrix}^{\top}\begin{bmatrix}\boldsymbol{L}_{\mathcal{X}\mathcal{X}} & \boldsymbol{L}_{\mathcal{X}\mathcal{Q}}\\ \boldsymbol{L}_{\mathcal{Q}\mathcal{X}} & \boldsymbol{L}_{\mathcal{Q}\mathcal{Q}}\end{bmatrix}\begin{bmatrix}\boldsymbol{\Psi}_{\mathcal{X}}\\ \boldsymbol{\Psi}_{\mathcal{Q}}\end{bmatrix} = \mathbf{0}.$$

Our problem is to find the label of the queries $\boldsymbol{\Psi}_{\mathcal{Q}}^{*} \approx \boldsymbol{\Psi}_{\mathcal{Q}}$, then we can find the prediction as follows:

$$\boldsymbol{\Psi}_{\mathcal{Q}}^{*} = -\left((\boldsymbol{L}_{\mathcal{Q}\mathcal{Q}})^{-1}\boldsymbol{L}_{\mathcal{Q}\mathcal{X}}\boldsymbol{\Psi}_{\mathcal{X}}^{\top}\right)^{\top}. \tag{4.11}$$

Finally, the model parameters Θ are updated via the prediction $\boldsymbol{\Psi}_{\mathcal{Q}}^{*}$ as follows:

$$\Theta = \arg\min_{\Theta}\left\|\boldsymbol{\Psi}_{\mathcal{Q}} - \boldsymbol{\Psi}_{\mathcal{Q}}^{*}\right\|^2. \tag{4.12}$$

In short, the multi-label propagation cell, upon receiving the support and query sets, constructs a graph and generates the concept scores of the query samples using Eq.(4.6). The concept scores, as mentioned before, are depicting the belief of the network regarding the classes that exist in the queries. Algorithm 3 details out the training steps of MLPN.

Algorithm 3 MLPN's Training Procedure

Input: Episodes T_i

$\Theta_0 \leftarrow$ random initialization
for t in $\{T_1, ..., T_{N_T}\}$ **do**
  $\mathcal{S} \leftarrow f_\Theta$(support images)
  $\mathcal{Q} \leftarrow f_\Theta$(query images)
  Normalize labels and form a matrix $\mathbf{\Psi}_\mathcal{X}$
  Construct $\mathcal{N}_i$ for each $\{\mathcal{X}, \mathcal{Q}\}$ as in Remark 1
  Construct a graph matrix $\boldsymbol{W}$ using Eq. 4.2
  Compute $\boldsymbol{L}$ using $\mathbf{I} - \boldsymbol{W}$
  Get concept scores $\mathbf{\Psi}^*_\mathcal{Q}$ using Eq. 4.6
  Compute final loss $\mathcal{L}_t$ using Eq. 4.4
  Update Θ using $\nabla_\Theta \mathcal{L}_t$
end for

Threshold. The estimation of $\mathbf{\Psi}^*$ is ensured to have positive element and unit ℓ_1 norm on the columns. Afterwards, a label set can be predicted via thresholding. Our idea here is to learn the threshold using an MLP. In particular, the estimated $\mathbf{\Psi}_\mathcal{Q}{}^*$ is utilized to estimate a soft-threshold τ by minimizing,

$$\mathcal{L}_{Thres} = -\sum_{j=1}^{|\mathcal{C}|} \boldsymbol{y}_{(j)} \log(\sigma_j) + (1 - \boldsymbol{y}_{(j)}) \log(1 - \sigma_j), \tag{4.13}$$

where σ_j denotes the Sigmoid function over the difference of the predicted value and a soft-threshold $(\mathbf{\Psi}^*_{(j)} - \tau)$. In our experiments, we use an MLP with 4 hidden layers and ReLU activation.

Inference. Note that the networks do not benefit of a set of queries in the testing stage. We tested the query one by one and obtaining the label estimation by computing:

$$\widetilde{\mathbf{\Psi}}_{\mathcal{Q}_j} = -\boldsymbol{L}_{\mathcal{Q}\mathcal{X}} \mathbf{\Psi}_{\mathcal{X}_j}. \tag{4.14}$$

Computational Complexity. The computational complexity of this approach for training the model is $\mathcal{O}(N^3_\mathcal{Q} + N_\mathcal{X} N_\mathcal{Q})$, where $N_\mathcal{X}$, $N_\mathcal{Q}$ are the number of the support set and the number of queries in an *episode*. The most expensive cost is the matrix inverse that can take $N^3_\mathcal{Q}$. However, the number of queries is not large to make the performance significantly slower than linear operation.

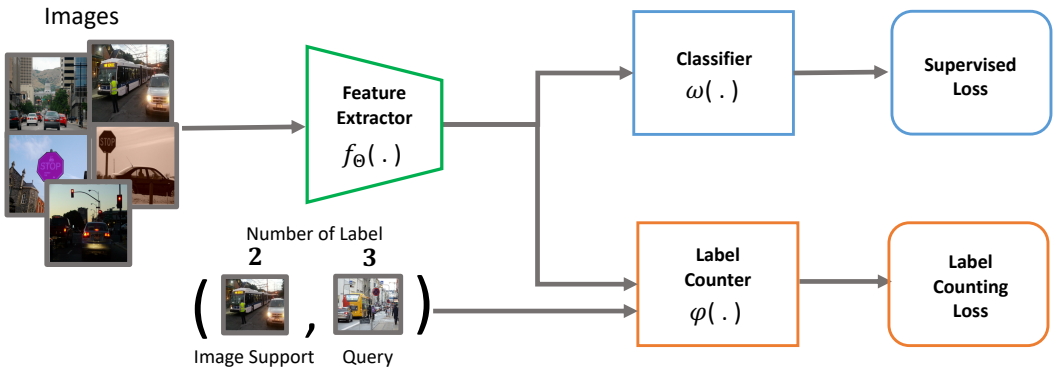


Figure 4.5: The overall architecture (training stage) including the classifier and the predictor of the number of labels.

4.3.1 Neural Label Count

We begin with the role of self-supervision learning for the multi-label problem. The characteristic of self-supervision training is to use the characteristic of the data. For instance, by rotating an image, one can self-supervise a machine by predicting the amount of rotation. Here, we make use of the arithmetic operation for prediction. The functionality of the NLC module is self-explanatory. The module predicts the number of classes (*e.g.*, objects in an image) presented in a given input.

Our design benefits from a relation module that takes into account a support sample, the query, and the global information of a task, represented in the support set. The feature of an input image and the feature of a query image are concatenated to obtain the total label prediction as shown in Figure 4.6. from The function to predict the number of labels is denoted by:

$$M^{[x_i,q]} = \varphi(f_\Theta(\boldsymbol{x_i}), f_\Theta(\boldsymbol{q}), \boldsymbol{z}), \tag{4.15}$$

where the multi-label counter function $\varphi : \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^{2|C|}$ learns the relation between two samples (*i.e.*, $f_\Theta(\boldsymbol{x_i})$ and $f_\Theta(\boldsymbol{q})$). Here, $\boldsymbol{z}$ is a vector carrying the context of the whole support set to the NLC module. In our implementation, we realize this as $\boldsymbol{z} = \frac{1}{NK}\sum_{x_i \in \mathcal{X}} f_\Theta(\boldsymbol{x_i})$.

In particular and as becomes clear shortly, the NLC predicts the number of objects presented collectively in $\boldsymbol{x_i}$ and $\boldsymbol{q}$. In doing so, the NLC module uses a softmax with $2|C|$ outputs as the maximum number of classes presented collectively in $\boldsymbol{x_i}$ and $\boldsymbol{q}$ cannot exceed $2|C|$.

Our training objective is to minimize the following loss function:

$$\mathcal{L}_E = \mathcal{L}_{su} + \lambda \mathcal{L}_{co}, \tag{4.16}$$

where $\mathcal{L}_{su}$ depends on the selection of the classifier and $\mathcal{L}_{co}$ is formulated as:

$$\mathcal{L}_{co} = -\sum_j \sum_i \log\left(\frac{\exp(M_{(j)}^{[x_i,q]})}{\sum_{j'} \exp(M_{(j')}^{[x_i,q]})}\right). \tag{4.17}$$

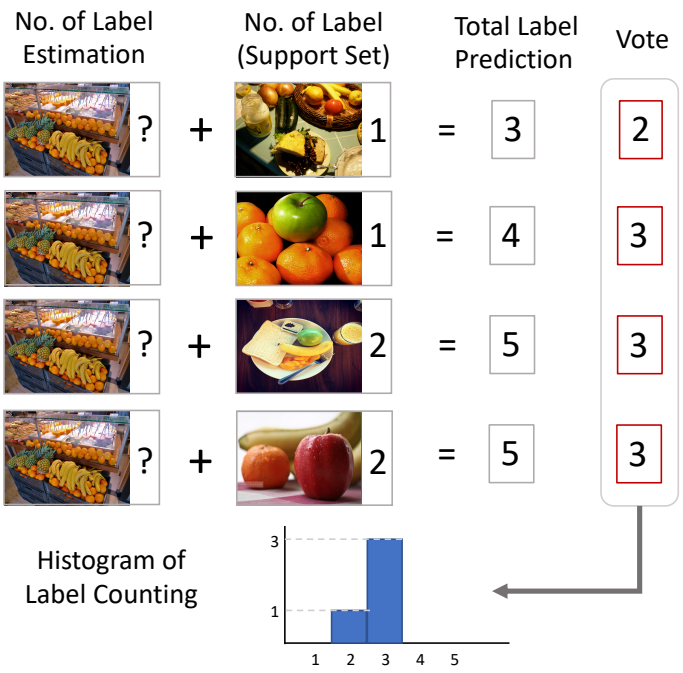


Figure 4.6: Estimating the number of labels by voting system of the support samples and queries. Here, an estimation (top) predicts the number of labels as 2 but the rest predictions are 3.

4.3.2 Inference with Label Count Voting

We employ a voting scheme on the output of the NLC module. To be more specific, we compare each support sample with the query and create the histogram of the label count estimation as shown in Fig. 4.6. The label counting based on histogram $\boldsymbol{H}$ is defined as:

$$\boldsymbol{H} = \{h(m)|0 \leq m \leq 2|C|\}, \tag{4.18}$$

with

$$h(m) = \sum_{\boldsymbol{x}_i \in \mathcal{X}} \mathbb{I}((M^{[\boldsymbol{x}_i,\boldsymbol{q}]} - B^{\boldsymbol{x}_i}) == m), \tag{4.19}$$

where $B^{\boldsymbol{x}_i}$ is the label count of the support sample $\boldsymbol{x}_i$ and $\mathbb{I}(.)$ returns 1 if the condition in its argument holds and 0 otherwise. Indeed, $h(m)$ shows how many times the number of classes in the query is counted as m. A majority voting is then used to make the final call as:

$$l = \underset{i \in \{1,\cdots,|C|\}}{\arg\max}\ h(i). \tag{4.20}$$

This label count estimation is also included in our evaluation scheme.

Remark 3. *We consider the episode style for training and testing multi-label few-shot classification. In this setting, the labels per episode are randomly picked and every episode may contain different samples. Thus, there is a chance that some objects are labeled in an episode*

but they are not labeled in another episode. This is also known as missing labels in multi-label classification. Based on this problem, the relation module is designed to be an adaptive module that captures the information based on the context in an episode.

Remark 4. *The NLC module has a different use compared to relation networks [Sung et al., 2018] that exploits the similarity between two features. The neural label count module conveys the operator relationship (*i.e.*, , summation) such that the result of this relation module is the number after an arithmetic operation.*

Remark 5. *This inference method makes use of the ensemble strategy to estimate the label count. The ensemble of label count estimations implies that a level of robustness might be expected. That is, even if one of the classifiers is wrong in its prediction, the other predictions may correct it through consensus. The final prediction is made based on the most frequent label count predictions. In the case of a tie, then the original predictions (floating point) are used to check the number with the highest probability.*

4.4 From Single-Label to Multi-Label Few-Shot Learning

Below, we extend and adapt prototypical networks [Snell et al., 2017] and relation networks [Sung et al., 2018].

Prototypical Networks. Prototypical networks learn a mapping $f_\Theta : \mathbb{R}^m \rightarrow \mathbb{R}^n$ from an input space to an embedding space. The main assumption is that the embeddings should sit near their class prototypes. Let us denote the prototypes (for C classes in an episode) by $\boldsymbol{P} = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_C\}; \boldsymbol{p}_k \in \mathbb{R}^n$ and $\boldsymbol{X}_i$ be a class-specific set. A prototype is calculated as the mean of the feature vectors that share the same class label:

$$\boldsymbol{p}_k = \frac{1}{|C_k|} \sum_{\boldsymbol{x}_j \in \boldsymbol{X}_k} f_\Theta(\boldsymbol{x}_j). \tag{4.21}$$

In prototypical networks, a query feature is classified by assigning it to the nearest prototype according to the Euclidean distance: $\boldsymbol{z}_{(j)} = \|\boldsymbol{p}_j - f_\Theta(\boldsymbol{q})\|^2$.

We adapt prototypical networks to work in the multi-label setting. For a given class, all samples annotated with that class are grouped together to form a prototype. Note that every sample with more than one label will contribute information of several prototypes. To perform a multi-label classification, we conduct training with a softmax function [Yang et al., 2016; Wang et al., 2017] in the final layer of a network. The objective function is formulated as follows:

$$\mathcal{L}_{PN} = \sum_{j=1}^{|\mathcal{C}|} \left(\frac{\boldsymbol{y}_{(j)}}{\|\boldsymbol{y}\|_1} - \frac{\exp(-\boldsymbol{z}_{(j)})}{\sum_{j'=1}^{|\mathcal{C}|} \exp(-\boldsymbol{z}_{(j')})} \right)^2. \tag{4.22}$$

Relation Networks. Few-shot recognition can also be performed via the use of the so-called *relation module* from relation networks (RN). This approach can be viewed as learning deep non-linear metric. By and large, RN consists of an embedding module and a *relation module*. Specifically, an embedding module is a non-linear mapping from the input space to a feature space $f_\Theta : \mathbb{R}^m \rightarrow \mathbb{R}^n$ and the *relation module* (g_Φ) learns the similarity between the query ($\boldsymbol{q}$)

Figure 4.7: Sample images from three datasets: MS-COCO, iMaterialist, and Open MIC.

and sample (x_j) in the support set. Let us denote X_i be a class-specific set. Given a problem with $\mathcal{C}$ classes, the relation scores can be calculated according to:

$$r_{(j)} = g_\Phi(f_\Theta(q), \frac{1}{|C_j|} \sum_{x_i \in X_j} f_\Theta(x_i)), \quad j = 1, \ldots, \mathcal{C}. \tag{4.23}$$

Architecture-wise, the *relation module* consists of two convolutional blocks, two fully connected layers, and a sigmoid function (σ). Training is performed by calculating mean squared error between the score and the query label w.r.t. the model parameters and embeddings:

$$(\Theta, \Phi) = \arg\min_{\Theta,\Phi} \sum_{j=1}^{|\mathcal{C}|} (r_{(j)} - y_{(j)})^2. \tag{4.24}$$

Thus, RN passes the features from the support set and a query to the binary classifier and the label with highest score is chosen as the predicted class.

Adapting RN from single-label to multi-label setting is straightforward. The *relation module* acts as a non-linear metric, thus, we can directly use a log-loss w.r.t. $r_{(j)}$ and labels y_j:

$$\mathcal{L}_{RN} = \sum_{j=1}^{|C|} y_{(j)} \log(r_{(j)}) + (1 - y_{(j)}) \log(1 - r_{(j)}). \tag{4.25}$$

4.5 Experiments

4.5.1 Datasets

We use three datasets detailed below to evaluate the capability of the methods to conduct ML-FSL. Note that, we have two disjoint sets for training and testing, so the images with annotated classes in the testing set do not appear in the training set. We propose the new splits for these three datasets. Fig. 4.7 shows examples from all datasets.

MS-COCO [Lin et al., 2014]. The MS-COCO dataset is built for object detection and recognition, thus, the data is suitable for multi-label recognition. This dataset comprises 80 classes

Model	Base Classes				Novel Classes			
	1-shot		5-shot		1-shot		5-shot	
	mAP	LC	mAP	LC	mAP	LC	mAP	LC
Pre-trained + MLP	56.6	-	57.5	-	50.2	-	54.4	-
Proto Nets	61.0	-	69.7	-	56.7	-	66.7	-
Relation Nets	64.4	-	69.3	-	57.3	-	63.3	-
MLPN	64.8	-	71.8	-	58.5	-	68.3	-
Proto Nets + NLC	62.8 ↑	40.2	72.3 ↑	45.1	58.0 ↑	**49.5**	68.0 ↑	54.1
Relation Nets + NLC	**66.2** ↑	**42.7**	71.0 ↑	42.8	59.0 ↑	48.0	66.1 ↑	**57.0**
MLPN + NLC	66.0 ↑	39.0	**74.5** ↑	**46.0**	**60.4** ↑	47.4	**69.1** ↑	54.6

Table 4.1: The accuracy (%) of baseline methods and the additional NLC on MS-COCO. The evaluation is based on mAP and LC for 10-way 1-shot and 5-shot. The best performance is in **Bold** and the second best is in Blue.

Model	1-shot	5-shot
LASO (intersection aug.)	40.5	57.2
LASO (union aug.)	45.3	58.1
Proto Nets	48.7	59.9
Relation Nets	49.5	58.5
MLPN	56.1	63.4
Proto Nets + NLC	**50.2**	**60.4**
Relation Nets + NLC	**53.3**	**60.8**
MLPN + NLC	**56.8**	**64.8**

Table 4.2: A comparison in mAP(%) to the existing benchmark [Alfassy et al., 2019] on 16-way 1-shot and 5-shot.

and 122,000 images in total. We split the training, validation, and testing set into 50, 10, and 20 classes, respectively. We categorize 50 training classes as base classes and 20 testing classes as novel classes. All of the images within training are disjoint from validation and testing sets. Additionally, we remove images that have less than two labels yielding 74,655 images. We use all of the images from the validation set of MS-COCO to evaluate performance on base classes.

In addition, we also evaluate on the MS-COCO split in [Alfassy et al., 2019] to compare with the existing approach. This split has 64 and 16 classes for training and testing, respectively.

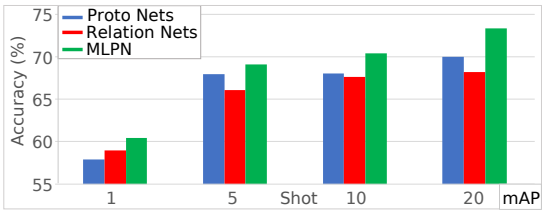


Figure 4.8: The impact of shot on MS-COCO (mAP).

Model	15-way			
	1-shot		5-shot	
	mAP	LC	mAP	LC
Proto Nets	48.4	-	59.8	-
Relation Nets	52.2	-	55.9	-
MLPN	52.3	-	61.1	-
Proto Nets + NLC	49.4 ↑	36.5	61.0 ↑	**45.0**
Relation Nets + NLC	**54.4** ↑	**41.1**	54.4 ↑	35.7
MLPN + NLC	53.6 ↑	38.1	**63.6** ↑	43.6

Table 4.3: The accuracy (%) of baseline methods with and without the auxiliary NLC loss on the MS-COCO. The evaluation is based on mAP and LC for 15-way 1-shot and 15-way 5-shot protocols. The best performance is highlighted by the **bold** font and the second best by the blue font.

Model	15-way			
	1-shot		5-shot	
	mAP	LC	mAP	LC
Proto Nets	52.9	-	62.6	-
Relation Nets	58.4	-	64.6	-
MLPN	56.4	-	59.2	-
Proto Nets + NLC	54.0 ↑	28.0	64.1 ↑	**31.9**
Relation Nets + NLC	**59.9** ↑	27.7	**65.3** ↑	31.9
MLPN + NLC	57.4 ↑	**29.5**	61.0 ↑	29.5

Table 4.4: The accuracy (%) of baseline methods without and with the auxiliary NLC loss on iMaterialist. The evaluation is based on mAP and LC for the 15-way 1-shot and 15-way 5-shot protocols. The best performance is highlighted by the **bold** font and the second best by the blue font.

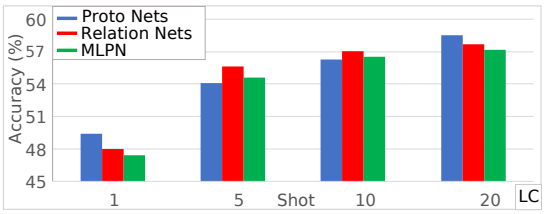


Figure 4.9: The impact of shot on MS-COCO (LC).

iMaterialist. We evaluate our ML-FSL proposal on the iMaterialist fashion dataset [Guo et al., 2019] consisting more than 1 million images and 228 distinct labels. We divide the dataset into training, validation, and testing sets for our purpose. We consider the subset of the dataset of 120 labels from all labels. The splits for testing and validation are 40 and 15, respectively. The rest labels are used for training. This fashion dataset shows the multi-label problem that has multiple labels for one object. For example, a color and a texture share the same visual

Model	1-shot		5-shot	
	mAP	**LC**	**mAP**	**LC**
Proto Nets	60.8	-	66.4	-
Relation Nets	62.1	-	67.4	-
MLPN	62.3	-	65.2	-
Proto Nets + NLC	62.6 ↑	32.2	68.9 ↑	35.1
Relation Nets + NLC	**64.0** ↑	39.0	**69.0** ↑	35.7
MLPN + NLC	63.5 ↑	**42.4**	66.8 ↑	**37.0**

Table 4.5: The accuracy (%) of baseline methods and the additional NLC on iMaterialist. The evaluation is based on mAP and LC for 10-way 1-shot and 5-shot. The best performance is in **Bold** and the second best is in Blue.

Model	1-shot				5-shot			
	$p1 \rightarrow p2$	$p2 \rightarrow p3$	$p3 \rightarrow p4$	$p4 \rightarrow p1$	$p1 \rightarrow p2$	$p2 \rightarrow p3$	$p3 \rightarrow p4$	$p4 \rightarrow p1$
Pre-trained + MLP	62.24	52.85	66.89	51.12	83.93	74.93	86.93	70.63
Proto Nets	58.48	54.82	67.41	50.89	81.52	75.71	86.07	72.64
Relation Nets	48.50	48.74	63.03	50.93	53.57	66.26	72.50	57.20
MLPN	**65.41**	**60.06**	**74.60**	**53.02**	**91.34**	**79.57**	**90.89**	**77.15**

Table 4.6: ML-FSL results (mAP) on Open MIC for 10-way 1-shot and 5-shot. $p\{n\} \rightarrow p\{m\}$ means that training is performed in *p{n}* and testing is applied in *p{m}*. The best performance is in **Bold** and the second best is in Blue.

appearance but they have different labels.

Open MIC [Koniusz et al., 2018]. This dataset contains images collected from 10 museum exhibition spaces. There are 866 classes in total and every class comprises 1-20 images per class. The images suffer from various photometric and geometric distortions. Multi-label annotations are available as every image contains more than one exhibit. The dataset is split into four subsets: *p1=(shn+hon+clv)*, *p2=(clk+gls+scl)*, *p3=(sci+nat)*, *p4=(shx+rlc)*.

4.5.2 Details

Implementation. We equip all methods in our experiments with the same variant of convolutional neural networks (CNN). The CNN has 4-convolutional layers (4-Conv) with 64 filters in each layer followed by batch normalization, ReLU, and max-pooling. Training and testing is performed over 100,000 and 1,000 episodes for both architectures. Adam optimizer [Kingma and Ba, 2015] with learning rate 0.001 is used whose learning rate is reduced by half every 10,000 episodes. We use $\lambda = 0.01$ for all methods and datasets.

We perform 10-way 1-shot and 10-way 5-shot experiments to compare the performance. The number of query images is half of the number of sampled labels. On MS-COCO, the testing stage employs a model from training for which the best validation score was attained. On the Open MIC dataset, we employ the models saved on the last training *episode*.

Evaluation Metric. The evaluation metrics used in our experiments are based on precision (P),

Table 4.7: The accuracy (%) which quantifies multi-label hard predictions. The counted label is used in making predictions based on thresholds.

Model	MS-COCO				iMaterialist			
	1-shot		5-shot		1-shot		5-shot	
	10-way	15-way	10-way	15-way	10-way	15-way	10-way	15-way
Proto Nets + NLC	30.4	20.9	37.1	24.8	45.6	**43.6**	47.4	**47.2**
Relation Nets + NLC	26.2	14.3	29.2	17.27	**46.5**	42.7	**49.7**	47.0
MLPN + NLC	**31.1**	**24.2**	**37.8**	**28.42**	45.4	40.4	48.9	43.5

recall (R), and F1 score defined as follows:

$$\mathrm{P} = \frac{N^c}{N^p} \quad \mathrm{R} = \frac{N^c}{N^g} \quad \mathrm{F1} = \frac{2P \times R}{P + R}, \tag{4.26}$$

where N^p, N^c, N^g denote the number of predictions, correct predictions, and ground truth, respectively. We report the precision and recall results for top $k = 3$ labels per image and a preset threshold to obtain predictions. Another measurement is the accuracy to estimate the label count (LC).

4.5.3 Results and Ablation Studies

In our experiments, we consider the multi-label classification problem in the meta-learning setting to explore the assumption that the tasks given during training and testing are similar. The experiments are performed with baselines on the 4-Conv backbone. In addition to the baselines, we perform a comparison to the pre-trained feature extractor on the training set and a multi-layer perceptron (MLP) layer as a classifier which is updated based on the given support set.

MS-COCO. In this dataset, we evaluate the methods on the base classes and the novel classes. This protocol makes sure that the model does not only fit to the novel classes and *forget* the base classes. We can observe the results from Table 4.1 that pre-trained feature extractor with MLP cannot perform better than the few-shot learning baselines trained from scratch with episodic training. In all cases, Multi-Label Propagation Network (MLPN) can outperform the other methods. MLPN outperforms the second highest on novel classes by 1.4% and 1.1% for 1-shot and 5-shot, respectively. This is because a graph model can capture the relation among samples which share similarity. Furthermore, there are also improvements for both base and novel classes when the basic few-shot learning methods are complemented with NLC. We conjecture that NLC imposes a regularization implicitly. On MS-COCO split by [Alfassy et al., 2019], we outperform the LASO model by 10% and 6% for 16-way 1-shot and 16-way 5-shot, respectively (see Table 4.2).

The Effect of Shot. We investigate the effect of the number of samples for mAP and LC. It is clearly shown in Fig. 4.9 that more additional samples are beneficial to estimate the label count. The baselines are also improved when more data is given as shown in Fig. 4.8. We can observe that mAP and LC increase about 10% or more from 1-shot to 20-shot.

iMaterialist. In this dataset, the data has a different structure compared to MS-COCO and

Open MIC because two labels can share the same object and the labels are hierarchical. Adding a label count loss is consistently beneficial for improving the accuracy (mAP). We observe a further improvement $\sim$1.5% with NLCfor 10-way 1-shot and 5-shot as shown in Table 4.5. Relation Nets [Sung et al., 2018] has the highest performance for 10-way 1-shot and 5-shot. Our conjecture is that the fashion images have less tight relationship between one label to another label than images in MS-COCO. For instance, a red color can appear to any fashion entities such as dress, trouser, or shirt but it is unlikely that an apple can appear on images containing sport equipment.

Open MIC. On the Open MIC dataset, the evaluation is performed on 4 different exhibitions for 10-way 1-shot and 5-shot. In all cases, MLPN outperforms by significant margins compared to the other three baselines with a minimum of 3% in mAP as shown in Table 4.6.

4.6 Chapter Conclusion

In this chapter, we introduce a meta-learning framework for multi-label few-shot classification and deep embedding with multi-label propagation cells. The meta-learning is given in the form of *episode* such that the models can gain experience by learning from past tasks and generalize to new tasks which are similar in their nature to the past tasks. We have demonstrated that meta-learning is feasible for the problem of multi-label few-shot classification. To this end, we have measured the performance on three challenging datasets: MS-COCO, iMaterialist, and Open MIC.

Apart from multi-label few-shot learning protocols, we have proposed deep embedding networks called multi-label propagation networks. Specifically, our proposed model comprises the deep embedding, multi-label propagation cell, and a classifier. The empirical results demonstrate that we attain improvements in FSL given our proposed model and we outperform other state-of-the-art methods in fair comparisons. There is also a possible improvement by considering the co-occurrence of multi-labels for multi-label propagation networks for future endeavor.

Chapter 5

On Modulating the Gradient for Meta-Learning

In this chapter, we investigate the problem of fast-adaptation in a low-data regime. The idea is to extend the gradient-based meta-learning method to quickly update the parameters when learning a new task. Our proposed algorithm uses a non-linear function to modulate the gradient, also well-known as a preconditioner. The empirical results show that our proposed method can achieve faster convergence while also improving the performance over prior gradient-based methods.

5.1 Introduction

Gradient-based algorithms [Finn et al., 2017; Zintgraf et al., 2019; Rusu et al., 2019; Ravi and Larochelle, 2017; Antoniou et al., 2019; Rajeswaran et al., 2019] can be successfully employed to address a broad set of problems in meta-learning including image classification [Sun et al., 2019a; Antoniou et al., 2019], regression [Zintgraf et al., 2019; Finn et al., 2017], and reinforcement learning [Al-Shedivat et al., 2018] to name a few. Despite the success, employing gradient-based methods in the absence of abundant data (*e.g.*, few-shot learning) is daunting as gradients become *noisy* as described in [Zhang et al., 2019]. To outline the setup, consider training a neural network with parameters $\boldsymbol{\theta} \in \Theta$. Let

$$\mathcal{L}_N(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \ell(\boldsymbol{x}_i, y_i, \boldsymbol{\theta}),$$

be the total loss of the network where $\ell : \mathcal{X} \times \mathcal{Y} \times \Theta \rightarrow \mathbb{R}^+$ denotes the loss for the input $\boldsymbol{x}_i \in \mathcal{X}$ and desired output $y_i \in \mathcal{Y}$. In majority of cases, training proceeds by computing the gradient $\boldsymbol{g} = \nabla_{\boldsymbol{\theta}} \mathcal{L}_N(\boldsymbol{\theta}) = \frac{\partial}{\partial \boldsymbol{\theta}} \mathcal{L}_N(\boldsymbol{\theta})$ followed by updates in the form

$$\boldsymbol{\theta}^{(k)} = \boldsymbol{\theta}^{(k-1)} - \alpha \nabla_{\boldsymbol{\theta}^{k-1}} \mathcal{L}_N\big(\boldsymbol{\theta}^{(k-1)}\big) \ . \tag{5.1}$$

Upon availability of ample data (a big enough N), updates using the gradient will hopefully converge to a good minimum. The dependency of the gradient on N (number of samples) immediately suggests the noisy nature of $\boldsymbol{g}$ in the low-data regime. That is, if we can only see

$n \ll N$ samples,

$$\hat{\boldsymbol{g}} = \nabla_{\boldsymbol{\theta}} \mathcal{L}_n(\boldsymbol{\theta}) = \frac{\partial}{\partial \boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^{n} \ell(\boldsymbol{x}_i, y_i, \boldsymbol{\theta}) \approx \boldsymbol{g} \,.$$

The issue intensifies if the learning algorithm benefits from the Hessian information or if the distribution of data changes. In meta-learning, the use of former might be tempting to achieve faster convergence while the latter occurs frequently due to the nature of the problem (*e.g.*, adaptation to new tasks).

To combat this issue, one would ideally like to use smaller learning rates for the noisy elements of the gradient, hence reducing their effects. This is indeed the underlying idea of some modern meta-learning algorithms, one way or another.

For example, the Meta-SGD [Li et al., 2017] algorithm explicitly formulates the learning problem as finding meta-parameters of the network along their optimal learning rate. In LEO [Rusu et al., 2019], updates are performed in a low-dimensional space and the result is consequently projected to the parameter space using a non-linear mapping.

A tool from optimization, called preconditioning, is a principled way for accelerating the convergence rate of the first-order methods. The idea is that instead of updates in the form of Eq. 5.1, one would use

$$\boldsymbol{\theta}^{(k)} = \boldsymbol{\theta}^{(k-1)} - \alpha \boldsymbol{g}\left(\boldsymbol{\theta}^{(k-1)}\right) \,. \tag{5.2}$$

Obviously, if the preconditioner is chosen to be the inverse of the Hessian matrix (*i.e.*, , $\boldsymbol{P} = \boldsymbol{H}^{-1}$ for $\boldsymbol{H} = \frac{\partial^2}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^\top}$), or approximates $\boldsymbol{H}^{-1}$ well enough for that matter, preconditioning reduces to the Newton method which enjoys a quadratic convergence rate. Inspecting Eq. 5.2 raises a question if it *is possible to use the idea of preconditioning to address the noisy gradient problem in meta-learning?* To answer this question, we hypothesize that in contrast to a full preconditioning matrix, using just the diagonal preconditioner may effectively alter the learning rate in an element-wise fashion and provide acceleration of the convergence of the learner. In this paper, we study this particular idea in detail. In particular and inspired by the concept of preconditioning, we make the following contributions.

5.1.1 Contributions

In this chapter, our contributions are:

i. We propose a meta-learning algorithm to learn to modulate the gradient in the absence of abundant data. Similarly to preconditioning and as the name implies, the gradient modulation is a multiplicative corrective factor albeit task-dependent. Being vigilant about the scalability, we formulate the modulation via low-rank approximation, which in turn lets us apply modulation on large models. Such an idea is significantly different from previous works which suffer from poor scalability and require adaptation of parts of the network (usually the classifier).

ii. We extensively compare and contrast our algorithm against state-of-the-art methods on several tasks, ranging from image classification to image completion to reinforcement

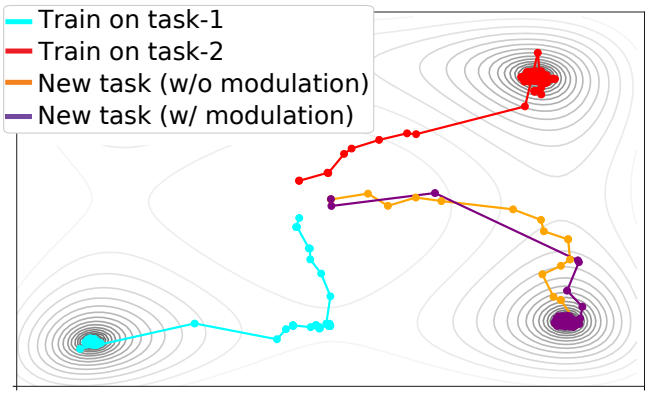


Figure 5.1: An illustration of our modulation applied for a new task after learning from two previous tasks (task-1 and task-2).

learning. Empirically, our method outperforms various state-of-the-art algorithms and exhibits faster convergence (see Fig. 5.1 for an illustration).

iii. We further study the robustness of the meta-learning algorithms in the presence of corrupted gradients. We observe that our idea of modulating the gradient is able to recover gracefully while other methods severely underperform.

5.2 Background

The objective of meta-learning is to **1.** achieve rapid convergence for new tasks (task-level) and **2.** generalize beyond previously seen tasks (meta-level). A common approach to meta-learning is to design models that learn from limited data using the concept of episodic training [Vinyals et al., 2016; Santoro et al., 2016]. Therein, a model is presented with a set of tasks (*e.g.*, image classification), where for each task, only limited data is available. To put the discussion into context, we first provide a brief overview of the Model Agnostic Meta Learning (MAML) algorithm [Finn et al., 2017]. Let $\mathcal{D}_\tau^{trn}$ and $\mathcal{D}_\tau^{val}$ be the training and the validation sets of a given task $\tau \sim p(\mathcal{T})$, respectively. We assume that $\mathcal{D} := \{\boldsymbol{x}_i, y_i\}_{i=1}^{|\mathcal{D}|}, \boldsymbol{x}_i \in \mathcal{X},\ y_i \in \mathcal{Y}$ for some small $|\mathcal{D}|$. Furthermore, let $h : \mathcal{X} \times \mathbb{R}^n \to \mathcal{Y}$ be the predictor function of the model parameterized by $\boldsymbol{\theta} \in \mathbb{R}^n$. The MAML algorithm seeks a universal initialization $\boldsymbol{\theta}^*$ by minimizing:

$$\min_{\boldsymbol{\theta}^*} \sum_{\tau \sim P(\mathcal{T})} \mathcal{L}\left(D_\tau^{\text{val}}, \boldsymbol{\theta}^* - \alpha \sum_{k=0}^{(K-1)} \nabla \mathcal{L}\left(D_\tau^{\text{trn}}, \boldsymbol{\theta}_\tau^{(k)}\right)\right). \tag{5.3}$$

Here, $\boldsymbol{\theta}_\tau^{(k)} = \boldsymbol{\theta}_\tau^{(k-1)} - \alpha \nabla \mathcal{L}\left(D_\tau^{\text{trn}}, \boldsymbol{\theta}_\tau^{(k-1)}\right)$ with $\boldsymbol{\theta}_\tau^0 = \boldsymbol{\theta}^*$. The loss terms are:

$$\begin{aligned} \mathcal{L}\left(D_\tau^{\text{trn}}, \boldsymbol{\theta}\right) &:= \mathbb{E}_{\boldsymbol{x},y \sim \mathcal{D}_\tau^{\text{trn}}}\left[\ell(h(\boldsymbol{x}, \boldsymbol{\theta}), y)\right], \\ \mathcal{L}\left(D_\tau^{\text{val}}, \boldsymbol{\theta}\right) &:= \mathbb{E}_{\boldsymbol{x},y \sim \mathcal{D}_\tau^{\text{val}}}\left[\ell(h(\boldsymbol{x}, \boldsymbol{\theta}), y)\right]. \end{aligned} \tag{5.4}$$

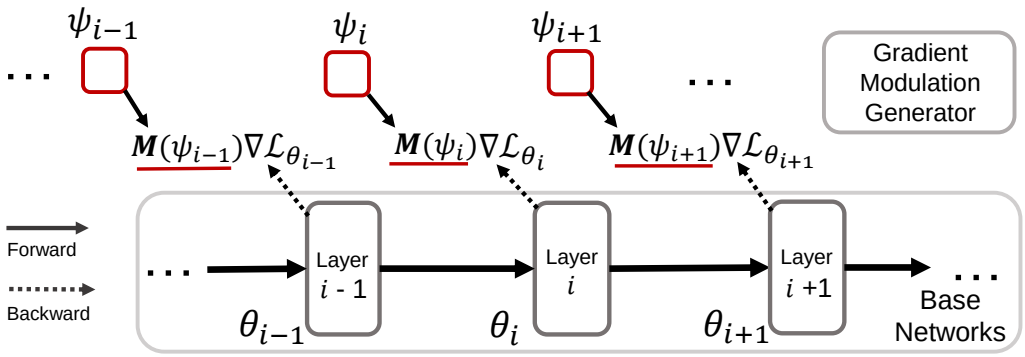


Figure 5.2: Every layer is equipped with a gradient correction generator to modulate the incoming gradients.

Intuitively, given a task τ, the MAML starts from $\boldsymbol{\theta}^*$ and performs K gradient updates on D_τ^{trn} to obtain the adapted parameters $\boldsymbol{\theta}_\tau^{(K)}$ (this is called the inner-loop updates). Then it uses D_τ^{val} and $\boldsymbol{\theta}_\tau^{(K)}$ (which is dependent on $\boldsymbol{\theta}^*$) to improve the universal initialization point $\boldsymbol{\theta}^*$ (this is called the outer-loop update). The extensions of MAML [Finn et al., 2017] include Meta-SGD [Li et al., 2017] and Meta-Curvature [Park and Oliva, 2019] with adaptive learning rates, CAVIA [Zintgraf et al., 2019] with feature modulation, and LEO [Rusu et al., 2019] with low dimensional updates.

5.3 Proposed Method

In this section, we introduce our meta-learner to accelerate the learning process for few- and multi-shot classification, regression, and reinforcement learning. Our proposed method essentially learns to **Mod**ulate the **Grad**ient via so-called meta-learning **ModGrad** such that each task has a specific gradient modulator depending on the context.

5.3.1 Inner-Loop with Gradient Modulation

We define $\boldsymbol{M} : \mathbb{R}^d \rightarrow \mathbb{R}^n$, a function with parameter $\boldsymbol{\Psi}$ that performs gradient modulation. The purpose of the gradient modulation is two-fold: **1.** to suppress the noise and **2.** accelerate the convergence by amplifying certain elements of the gradient. Inspired by the use of diagonal for preconditioning, we formulate the meta-learning algorithm as follows:

$$\min_{\boldsymbol{\theta}^*, \boldsymbol{\Psi}} \sum_{\tau \sim P(\mathcal{T})} \mathcal{L}\left(D_\tau^{\text{val}}, \boldsymbol{\theta}^* - \alpha \sum_{k=0}^{K-1} \boldsymbol{M}_\tau^{(k)}(\boldsymbol{\nu}, \boldsymbol{\Psi}) \odot \nabla \mathcal{L}\left(D_\tau^{\text{trn}}, \boldsymbol{\theta}_\tau^{(k)}\right)\right). \tag{5.5}$$

The inner-loop update in Eq. 5.5 performs an element-wise modulation of the gradient vector by operator '$\odot$'. One can also view this updating scheme as a generalization of meta-learners that adaptively alter the learning rate of the SGD (*e.g.*, [Antoniou et al., 2019; Li et al., 2017; Rusu et al., 2019]). In what follows next, we build a generative modulator through another neural network.

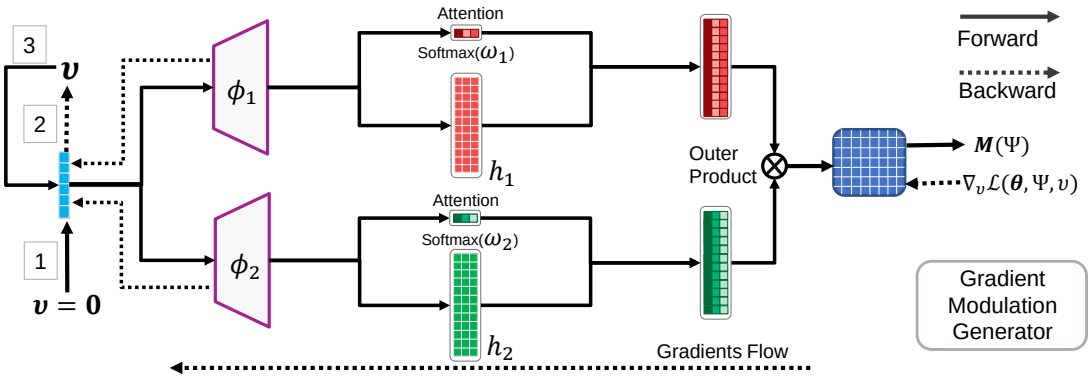


Figure 5.3: A gradient modulation generator. Two sister networks (ϕ_1, ϕ_2) produce two tall matrices to generate the correction.

5.3.2 Task-Dependent Gradient Modulation Generator

By minimizing Eq. 5.5, we jointly learn the universal initialization vector $\boldsymbol{\theta}^*$ and the generator $\boldsymbol{M}_\tau^{(k)}(\boldsymbol{\Psi})$ to enrich adaptability of the meta-learner. Fig. 5.2 and 5.3 illustrate our design to generate the modulation. Without loss of generality, we denote $n = n_1 \times n_2$ and we elaborate on how the function $\boldsymbol{M}_\tau^{(k)}(\boldsymbol{\Psi}) \in \mathbb{R}^n$ is obtained via the generator for a given task τ.

For reasons that become clear shortly, the generator makes use of a context vector $\boldsymbol{v} \in \mathbb{R}^d$ to generate the gradient modulation $\boldsymbol{M}_\tau^{(k)}(\boldsymbol{\Psi})$. In doing so, the context vector $\boldsymbol{v}$ is first processed by two sister modules ϕ_1 and ϕ_2. This generates,

$$\begin{aligned} (\boldsymbol{\omega}_1, \boldsymbol{h}_1) &= \phi_1(\boldsymbol{v}_\tau^{(k)}), \quad \boldsymbol{\omega}_1 \in \mathbb{R}^u, \boldsymbol{h}_1 \in \mathbb{R}^{n_1 \times u}, \\ (\boldsymbol{\omega}_2, \boldsymbol{h}_2) &= \phi_2(\boldsymbol{v}_\tau^{(k)}), \quad \boldsymbol{\omega}_2 \in \mathbb{R}^u, \boldsymbol{h}_2 \in \mathbb{R}^{n_2 \times u}. \end{aligned} \tag{5.6}$$

Attention mechanism is employed to re-weigh the matrix produced by two sister modules:

$$\begin{aligned} \boldsymbol{a}_1 &= \text{softmax}(\boldsymbol{\omega}_1), \\ \boldsymbol{a}_2 &= \text{softmax}(\boldsymbol{\omega}_2). \end{aligned} \tag{5.7}$$

The softmax function is chosen because we observe empirically that it always performs the best compared to other activation functions. The output of the gradient modulator is then computed by the outer product operation:

$$\boldsymbol{M}_\tau^{(k)}(\boldsymbol{\Psi}) = \text{Vec}\big((\boldsymbol{h}_1 \boldsymbol{a}_1) \otimes (\boldsymbol{h}_2 \boldsymbol{a}_2)\big). \tag{5.8}$$

$\text{Vec}(\cdot)$ operator vectorizes an input matrix. Eq. 5.8 uses a low-rank approximation to generate gradient corrections. This lets us scale up ModGrad to very large networks and simultaneously regularizes gradients. Finally, the produced matrix is passed via ReLU.

Finally, the only remaining detail of the ModGrad algorithm is the context vector generation. To this end, we firstly reset $\boldsymbol{v}_\tau^{(0)} = \boldsymbol{0}$ and then we generate an initial modulation $\boldsymbol{M}_\tau^{(k)}(\boldsymbol{\Psi})$. We then update $\boldsymbol{v}$ as:

$$\boldsymbol{\nu}_{\tau}^{(k)} = -\nabla_{\boldsymbol{\nu}_{\tau}^{(k-1)}} \mathcal{L}\big(\boldsymbol{\theta}^{(k-1)} - \alpha \boldsymbol{M}_{\tau}^{(k-1)}(\boldsymbol{\Psi}) \odot \nabla_{\boldsymbol{\theta}^{(k-1)}} \mathcal{L}(\boldsymbol{\theta}^{(k-1)})\big) \,. \tag{5.9}$$

The context vector $\boldsymbol{\nu}$ is then fed into two sister networks (Eq. 5.6) to compute the modulation. Algorithm 4, outlined for classification and regression tasks, provides details of how the parameters of the generator $\boldsymbol{\Psi}$ and the base network $\boldsymbol{\theta}$ are updated. For the reinforcement learning, another variant of ModGrad is provided in the supplementary material.

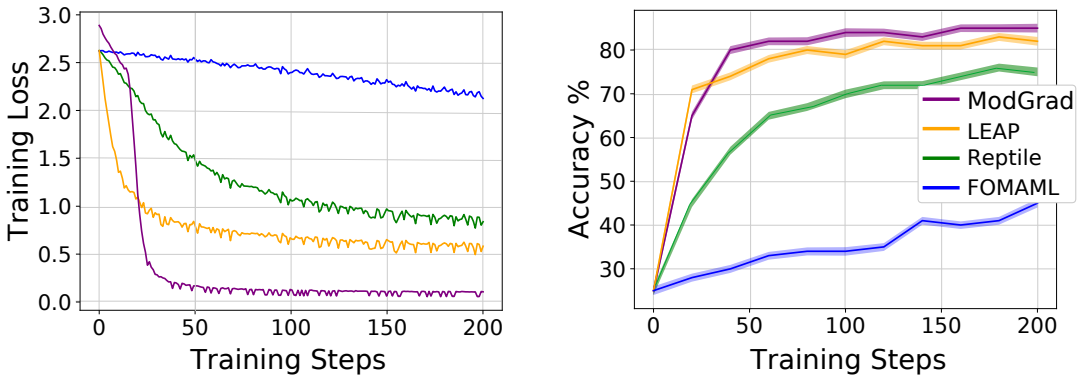
Algorithm 4 Train ModGrad

Require: $\boldsymbol{\theta}$, $\boldsymbol{\Psi}$, α, $p(\mathcal{T})$
$\boldsymbol{\theta}$, $\boldsymbol{\Psi} \leftarrow$ Random initialization
while not done **do**
  Sample $\tau_1 \dots \tau_B$ from $p(\mathcal{T})$ ▷ Sample episodes
  for b in $\{1, ..., B\}$ **do**
    $\boldsymbol{\theta}^0 \leftarrow \boldsymbol{\theta}$
    $\mathcal{D}_{\tau}^{trn}, \mathcal{D}_{\tau}^{val}$ from $\ll_b$ ▷ Sample training and testing sets
    for k in $\{1, ..., K\}$ **do**
      Reset $\boldsymbol{\nu}$ ▷ Reset context vector to 0
      Compute $\nabla_{\boldsymbol{\theta}^{(k-1)}} \mathcal{L}(\boldsymbol{\theta}^{(k-1)})$
      Compute $\boldsymbol{\nu}$ using Eq. 5.9 ▷ Update context vector
      Generate the gradient modulation using Eq. 5.8
      $\boldsymbol{\theta}^{(k)} \leftarrow \boldsymbol{\theta}^{(k-1)} - \alpha \boldsymbol{M}_{\tau}^{(k-1)}(\boldsymbol{\Psi}) \odot \nabla_{\boldsymbol{\theta}^{(k-1)}} \mathcal{L}(\boldsymbol{\theta}^{(k-1)})$
    end for
  end for
  $\boldsymbol{\theta} \leftarrow$ OptimizerStep($\mathcal{D}_{\tau}^{val}, \boldsymbol{\theta}^{(K)}$) ▷ Meta update base networks
  $\boldsymbol{\Psi} \leftarrow$ OptimizerStep($\mathcal{D}_{\tau}^{val}, \boldsymbol{\Psi}$) ▷ Meta update ModGrad
end while

Remark 6. *ModGrad uses a lower-dimensional context vector $\boldsymbol{\nu}$ to generate the gradient modulation. This enables us to lower the computational complexity even further. The underlying assumption, based on the smoothness of the gradient updates, is that the gradient field, especially for modern models which are often deep and over-parameterized, should comply with the low-dimensional latent data representation. As such, enforcing the context vector to be low-dimensional implicitly contributes to capturing the geometry of the gradient field.*

Remark 7. *In contrast to T-Net [Lee and Choi, 2018] and natural neural networks [Desjardins et al., 2015] that alter the neural networks by inserting additional layers for gradient projection, ModGrad directly produces the modulation vector for the gradients, thus it does not require altering the architecture of the base network to achieve adaptation.*

Remark 8. *Meta-SGD benefits from the adaptive learning rate to accelerate its convergence and improve the performance of standard MAML. In Meta-SGD, the modulator is a global parameter for all tasks. Our method differs from Meta-SGD in that we use a task-dependent gradient modulation generated from neural networks.*



(a) Training loss (b) Testing accuracy

Figure 5.4: Training loss and testing accuracy on Omniglot in the multi-shot setting.

Remark 9. *In practice and to lower the computational complexity, one can make use of a set of distinct ModGrad cells* $\boldsymbol{\Psi} = \{\boldsymbol{\psi}_1 \dots \boldsymbol{\psi}_m\}$*, each acting on and optimizing a layer of the network (see Fig. 5.2 for a conceptual diagram). A ModGrad cell may be combined with any part of a neural network including fully-connected and convolutional layers. This design requires no changes to the base network and brings flexibility, letting modulate the gradient for selected layers. Implementation details for fully-connected and convolutional layers are left in the supplementary material.*

5.4 Experiments

In this section, we compare and contrast the proposed ModGrad method against baselines and state-of-the-art algorithms on various meta-learning tasks such as few-shot classification, image completion, and robot navigation. We conclude the section by an ablation study and analysis of the robustness of ModGrad in the presence of noisy gradients.

5.4.1 Classification

Multi-shot classification. For multi-shot classification, we used the Omniglot dataset [Lake et al., 2015a] containing 1623 characters from 50 different alphabets. The task is to classify images given a random number of sampled images (class-imbalance). Following the multi-shot setting in [Flennerhag et al., 2019], we assess the rate of convergence and the performance of ModGrad in comparison to the LEAP [Flennerhag et al., 2019], Reptile [Nichol et al., 2018], and FOMAML [Finn et al., 2017] for a 20-way problem with 25 tasks in total (see Fig. 5.4 for details). We stress that in this experiment, the number of samples per class varies randomly between 1 and 25. Fig. 5.4 suggests that ModGrad performs adaptation much faster and to a much lower training loss while achieving the highest recognition accuracy of 85.1% compared to 83.3% of LEAP and 76.4% of Reptile. To the best of our knowledge, the performance of ModGrad for the multi-shot classification is the state of the art on this protocol and it tops previous studies by a notable margin.

Few-shot classification. As our next experiment, we benchmark the ModGrad method for few-shot classification on the *mini*-ImageNet dataset [Ravi and Larochelle, 2017]. In this comparison, we compare ModGrad to the related gradient-based meta-learning algorithms. The *mini*-ImageNet is a subset of the ImageNet [Krizhevsky et al., 2012] with 64, 16, and 20 classes for training, validation, and testing, respectively. We follow the widely-used protocol in the form of *episodes* for 5-way 5-shot and 1-shot with 600 tasks for testing. In all experiments, we performed episodic training using two network architectures, namely 4-convolutional blocks (Conv-4) without augmentation following [Snell et al., 2017; Finn et al., 2017] and WideResNet 28-10 (WRN-28-10) [Zagoruyko and Komodakis, 2016] with augmentation following [Qiao et al., 2018; Rusu et al., 2019].

As paper [Raghu et al., 2020] suggests that the representations in last layers undergo significant changes during adaptation, we apply updates to the last two convolutional layers of the base network. The model parameters of the base networks and ModGrad are optimized with the Adam optimizer [Kingma and Ba, 2015]. The learning rate is set to 10^{-3} and then cut by half for every 10K episodes. The size of ν and value of α are set to 300 and 0.1 for all experiments on *mini*-ImageNet.

On this benchmark, ModGrad outperforms existing few-shot methods for various backbones *e.g.*, Conv-4 and WRN-28-10 as presented in Table 5.1. For a fair comparison, the result is compared to the meta-learning algorithms with the same backbones and experimental setup. Using Conv-4, ModGrad only needs 1-step and 64 filters per layer to outperform CAVIA, which employs 5-steps and 512 filters, by around 1.4% and 3.3% for 5-way 1-shot and 5-shot protocols. Furthermore, ModGrad with WRN-28-10 also performs better than the works by [Lee et al., 2019b] and [Rusu et al., 2019]. ModGrad needs only 1-step in the inner-loop compared to other meta-learning methods that need more than 1-step to achieve good results.

Number of steps. Below, we investigate meta-learning on deeper networks using ResNet [He et al., 2016] and a higher number of shots to capture the relationship between the number of steps and these two factors. The *mini*-ImageNet dataset is used for experiments. To this end, we reimplement MAML and use the first-order method as the memory load for the second-order method is enormous given very deep networks. Note that the reported number of steps is applied for both training and testing stages for the 5-way classification. To investigate the relationship between the number of shot and the number of steps, the number of shot is set to 5, 10, 15, and 20 samples using the Conv-4 backbone. On ResNet-34, data augmentation and image size of 224×224 are used *without fine-tuning* the learning rate, following settings in [Chen et al., 2019a]. Training using Conv-4 and ResNet-34 is performed over 50K and 100K episodes, respectively. Fig. 5.5 shows that MAML [Finn et al., 2017] needs more steps to achieve better results for higher shot number and deeper networks.

On the ResNet-34, we observe that the performance gap for 5-steps and 30-steps on Conv-4 (64 filters) is 2.5% but the performance gap reaches 4% on ResNet-34. Using ResNet-34, ModGrad outperforms MAML by $\sim$6.5% and $\sim$2.5% for 5-way 1-shot and 5-shot protocols. Using higher shot numbers, MAML with more additional steps also shows the improvement. The performance gap for 5-shot is about 2.5% between 5-steps and 30-steps but the performance gap increases up to 4% for the 5-way 20-shot protocol. Furthermore, in 1-step, ModGrad outperforms 30-steps by MAML by 2% in 20-shot classification. We conjecture that ModGrad achieves a good performance in 1-step for both cases because the modulation adaptively scales

Model	Backbone	1-shot	5-shot
ML LSTM [Ravi and Larochelle, 2017]	Conv-4	43.44 ± 0.77	60.60 ± 0.71
MAML (64)$^{\#}$ [Finn et al., 2017]	Conv-4	47.89 ± 1.20	64.59 ± 0.88
Reptile [Nichol et al., 2018]	Conv-4	49.97 ± 0.32	65.99 ± 0.58
Meta-SGD ([Li et al., 2017]	Conv-4	50.50 ± 1.90	64.00 ± 0.90
R2-D2 [Bertinetto et al., 2019]	Conv-4	48.70 ± 0.60	65.50 ± 0.60
(M)T-Net [Lee and Choi, 2018]	Conv-4	51.70 ± 1.84	–
CAVIA (512) [Zintgraf et al., 2019]	Conv-4	51.82 ± 0.65	65.85 ± 0.55
ModGrad (1-step)	Conv-4	$\mathbf{53.20 \pm 0.86}$	$\mathbf{69.17 \pm 0.69}$
Qiao et al. [Qiao et al., 2018]	WRN-28-10	59.60 ± 0.41	73.74 ± 0.19
MTL [Sun et al., 2019a]	ResNet-12	61.20 ± 1.80	75.50 ± 0.80
LEO [Rusu et al., 2019]	WRN-28-10	61.76 ± 0.08	77.59 ± 0.12
SCA + MAML++ [Antoniou and Storkey, 2019]	DenseNet	62.86 ± 0.79	77.64 ± 0.40
wDAE-MLP [Gidaris and Komodakis, 2019]	WRN-28-10	62.67 ± 0.15	78.70 ± 0.10
wDAE-GNN [Gidaris and Komodakis, 2019]	WRN-28-10	62.96 ± 0.15	78.85 ± 0.10
Meta-Curvature [Park and Oliva, 2019]	WRN-28-10	64.40 ± 0.10	80.21 ± 0.10
ModGrad (1-step)	WRN-28-10	$\mathbf{65.72 \pm 0.21}$	$\mathbf{81.17 \pm 0.20}$

Table 5.1: The performance of existing gradient-based meta-learning methods for few-shot classification. Reported results are evaluated for 5-way 1- and 5-shot protocols on *mini*-ImageNet. MAML $^{\#}$ is our reimplementation.

the gradients to reach a good minimum rapidly.

5.4.2 Regression

The task of image regression (completion) on the CelebA dataset [Liu et al., 2015] is adopted from [Garnelo et al., 2018]. The goal is to in-paint missing image pixels given only some pixels of images (random and ordered). For inputs pixel locations, models have to perform regression to approximate pixel intensities given 10 and 100 training pixels. The results in Table 5.2 show that ModGrad has the lowest Mean Square Error (MSE) on the image regression tasks compared to Conditional Neural Process (CNP) [Garnelo et al., 2018], CAVIA [Zintgraf et al., 2019], and MAML [Finn et al., 2017]. We use the same setup as stated in [Zintgraf et al., 2019] with five 128 hidden layers and a 128-dimensional input vector (ν). The learning rate is set 0.1. For this regression task, ModGrad is applied only to a single fully-connected layer (preceding the last layer) as we observed no tangible difference if applying ModGrad to several layers. Note that our results use only 1-step while CAVIA [Zintgraf et al., 2019] and MAML [Finn et al., 2017] use 5 gradient steps to train from the training pixels. The qualitative results of image completion are shown in Fig. 5.6.

5.4.3 Reinforcement Learning

Our experiments on reinforcement learning are adopted from [Zintgraf et al., 2019; Finn et al., 2017]. All network architectures, range of parameters, and protocols follow the same setup.
2D Navigation. In this experiment, we evaluate ModGrad on 2D-Navigation tasks from [Finn et al., 2017]. Every task contains a randomly chosen goal position where an agent has to move

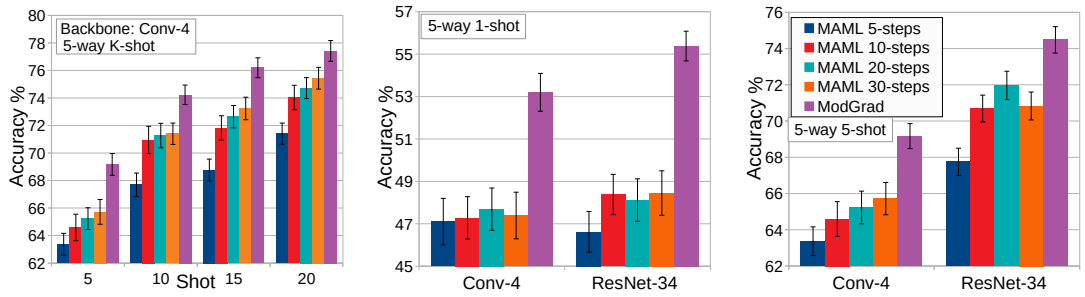


Figure 5.5: The performance of (a) ModGrad with various shot numbers, (b) deeper networks, and (c) MAML with 5, 10, 20, and 30 steps in the inner-loop. In 1-step, ModGrad achieves superior performance given various shots and backbones. MAML cannot perform well with a deeper network and larger shot numbers.

Model	Random Pixels			Ordered Pixels		
	10	**100**	**1000**	**10**	**100**	**1000**
CNP [Garnelo et al., 2018]	0.039	0.016	0.009	0.057	0.047	0.021
MAML [Finn et al., 2017]	0.040	0.017	0.006	0.055	0.047	0.007
CAVIA [Zintgraf et al., 2019]	0.037	0.014	0.006	0.053	0.047	0.006
ModGrad (1-step)	**0.034**	**0.012**	**0.005**	**0.048**	**0.043**	**0.005**

Table 5.2: The error rates for image completion tasks on 10, 100, and 1000 pixels on CelebA.

towards this position. The goal of this task is to adapt the policy of an agent quickly such that it can maximize the (negative) rewards. The goals of this navigation are within the range $[-0.5, 0.5]$ and the actions are clipped within $[-0.1, 0.1]$. In total, 20 trajectories are used for one gradient update. As in [Zintgraf et al., 2019], we use the same network with two-layers, 100 hidden units, and a ReLU activation function. In 1-step, ModGrad achieves rewards around -8 while CAVIA and MAML are far below with -15.

Locomotion. We evaluate our method on the half-cheetah locomotion tasks from the MuJoCo simulator [Todorov et al., 2012]. The tasks consist of predicting the direction and the velocity. The velocity ranges between 0.0 and 2.0. Each rollout length is 200, and 20 rollouts are used per gradient step during training. ModGrad reaches rewards around 590 with only 1-step but CAVIA and MAML obtain rewards below 550 only for half-cheetah direction tasks. Furthermore, ModGrad reaches around -80 for half-cheetah velocity tasks with 1-step but CAVIA and MAML reach only around -90 and -100, respectively.

In all reinforcement learning tasks,] Fig. 5.7 shows that ModGrad requires fewer updates to achieve better rewards. This shows that our method is also beneficial for non-differentiable and dynamic problems.

5.4.4 How Robust is ModGrad to Noisy Gradients?

Methods such as CAVIA [Zintgraf et al., 2019] and T-Net [Lee and Choi, 2018] modulate parameters or use an additional layer to transform gradients. These methods receive the gradients directly from the base network. Below, we empirically show that these approaches

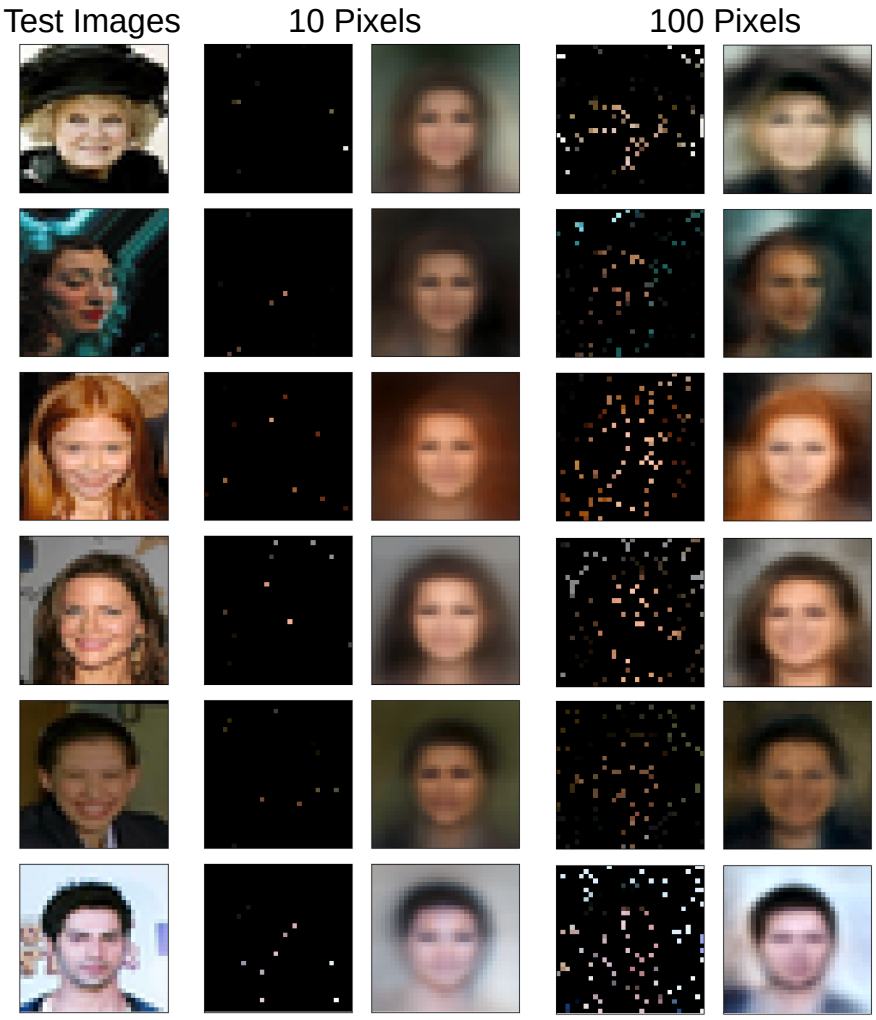
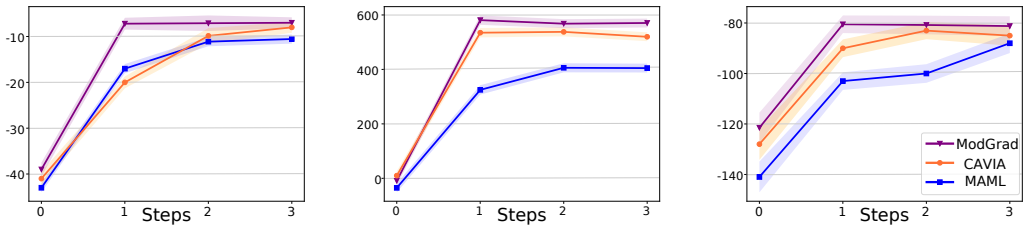


Figure 5.6: Qualitative results of ModGrad on the CelebA dataset for image completion with 10 and 100 pixels provided randomly.

are fragile in the presence of corrupted gradients (which is a fundamental issue in the few-shot regime). To evaluate the robustness, we corrupted the gradients $\nabla\mathcal{L}(\boldsymbol{\theta})$ by adding noise following $\mathcal{N}(\mathbf{0}_n, \eta\mathbf{1}_n)$ to the gradients in the inner-loop, and we measured the accuracy of ModGrad, CAVIA, T-Net for various η (see Fig. 5.8 for results) using Conv-4 on *mini*-ImageNet. Compared to other methods, ModGrad degrades gracefully. For example, while our method only degrades about 10%, T-Net, CAVIA, Meta-SGD, and MAML plummet by 30% and 40% for 5-way 1-shot and 5-way 5-shot protocols, respectively.

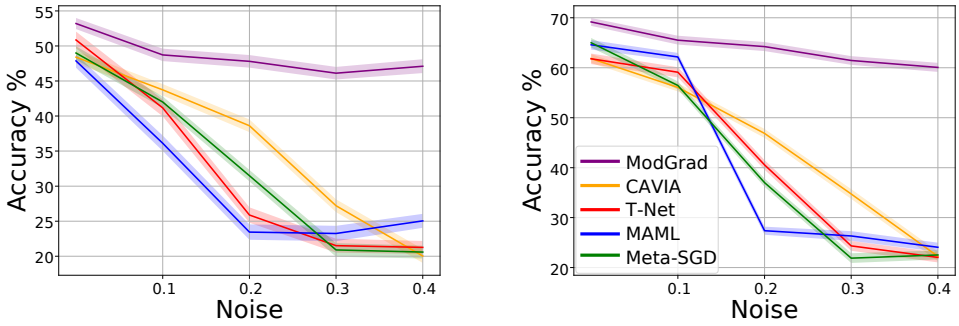
5.4.5 Ablation Study

Below, we provide an ablation study on our proposed method using Conv-4 backbone. The experiments show how the number of steps and the column dimension of the tall matrix (u) used by the outer product affect the performance.



(a) 2D Navigation (b) Half-Cheetah Dir (c) Half-Cheetah Vel

Figure 5.7: Results for reinforcement learning on 2D navigation, half-cheetah direction, and velocity.



(a) 5-way 1-shot (b) 5-way 5-shot

Figure 5.8: The performance comparison on *mini*-ImageNet with various noise level for 5-way 1-shot and 5-way 5-shot protocols.

Step	1	2	3	4	5
5-way 5-shot	69.17	68.80	68.42	68.34	68.24
5-way 1-shot	53.20	52.66	53.54	52.87	52.76

Table 5.3: ModGrad given various numbers of steps on *mini*-ImageNet.

Impact of steps. We run 5-steps in inner-loop to check the performance of 5-way 5-shot and 1-shot protocols on *mini*-ImageNet using Conv-4. Table 5.3 shows that ModGrad achieves a good performance in 1-step while running over more steps may vary the performance $\pm 1\%$ on 1-shot and 5-shot protocols. Thus, ModGrad is robust to a varying number of steps in few-shot learning.

The number of columns (u). Eq. 5.6 lets us choose the number of columns in the tall matrices. This experiment shows the impact of the number of columns (u) on results. Table 5.4 shows that the choice of u does not degrade the performance significantly ($\sim$0.5% variation on 5-way 1-shot protocol) but it may degrade results by $\sim$1.5% on the 5-way 5-shot protocol. In both cases, our approach outperforms the standard MAML (Conv-4) algorithm. We observe that the lower the value of u is the more the low-pass filtering nature of our gradient modulation is.

With a full-rank matrix, the modulator looses its low-pass filtering nature. Our experiments on classification, regression and reinforcement learning use the value of $u = 5$.

5.4.6 Discussion

Time Complexity. ModGrad requires two forward and two backward passes per step. Thus, its complexity is $2\times$ the computation of MAML but ModGrad converges significantly faster than MAML. For example, ModGrad with just one adaptation step comfortably outperforms MAML with five steps (best setting used by MAML). When comparing wall clocks, adaptation in MAML requires 0.05 second while ModGrad performs this step in 0.03 second per adaptation on the 5-way 5-shot protocol given Conv-4 backbone on mini-ImageNet.

Properties of ModGrad. We have observed that the low-rank modulating matrix paired with the Hadamard product acting on gradients have the property to perform adaptive low-pass filtering. This property depends on the context vector of the sister networks and the number of columns in the tall matrix controlled by the value of u. We believe this deems ModGrad a generative adaptive gradient filtering modulator which explains why ModGrad copes so well in the presence of gradients corrupted by the noise. Our supplementary material provides a more detailed theoretical analysis of the low-pass filtering properties of ModGrad. It also provides plots studying this property on both simulated and the real data.

5.5 Analysis of the modulator $M(\Psi, r) \odot \nabla\mathcal{L}$

Below, we analyze the properties of the ModGrad modulator and its robustness.

Denoising Property of Modulation. Our $\boldsymbol{M}(\boldsymbol{\Psi}, r) \in \mathbb{R}^{n}_{+}$ follows in fact an imposed row-column structure due to Eq. (8) (main manuscript). For brevity, we drop the input arguments of $\boldsymbol{M}$ and we assume it to be already of size $n_1 \times n_2 = n$, that is, $\boldsymbol{M} \in \mathbb{R}^{n_1 \times n_2}_{+}$. Furthermore, let gradient $\nabla\mathcal{L}(\cdot)$ (reshaped to $n_1 \times n_2$) be decomposed into a noise-free gradient term $\boldsymbol{G} \in \mathbb{R}^{n_1 \times n_2}$ and the noise matrix $\hat{\boldsymbol{\epsilon}} \in \mathbb{R}^{n_1 \times n_2}$ such that $\nabla\mathcal{L} = \boldsymbol{G} + \hat{\boldsymbol{\epsilon}}$. Finally, consider the SVD factorization of each matrix, that is, $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\lambda}\boldsymbol{V}^T$, $\boldsymbol{G} = \boldsymbol{U}'\boldsymbol{\beta}\boldsymbol{V}'^T$ and $\hat{\boldsymbol{\epsilon}} = \boldsymbol{U}^*\boldsymbol{\epsilon}\boldsymbol{V}^{*T}$. Then, consider the factorized version of the modulated gradient $\widetilde{\boldsymbol{G}} = \boldsymbol{M} \odot (\boldsymbol{G} + \hat{\boldsymbol{\epsilon}})$ given as:

$$\begin{aligned}\widetilde{\boldsymbol{G}} &= \sum_{i=1}^{r}\sum_{j=1}^{r'} \lambda_i \boldsymbol{u}_i \boldsymbol{v}_i^T \odot \left(\beta_j \boldsymbol{u}'_j \boldsymbol{v}'^T_j + \epsilon_j \boldsymbol{u}^*_j \boldsymbol{v}^{*T}_j\right) \\ &= \sum_{i=1}^{r}\sum_{j=1}^{r'} \left[\sqrt{\lambda_i}\boldsymbol{u}_i \odot \left(\sqrt{\beta_j}\boldsymbol{u}'_j + \sqrt{\epsilon_j}\boldsymbol{u}^*_j\right)\right]\left[\sqrt{\lambda_i}\boldsymbol{v}_i \odot \left(\sqrt{\beta_j}\boldsymbol{v}'_j + \sqrt{\epsilon_j}\boldsymbol{v}^*_j\right)\right]^T, \end{aligned} \tag{5.10}$$

Value of u	**1**	**5**	**10**
5-way 5-shot	68.43	69.17	67.62
5-way 1-shot	53.13	53.20	52.53

Table 5.4: The impact of column dimensions (u) on *mini*-ImageNet.

where $\odot$ is the Hadamard product, r and r' are the rank numbers for the modulator and the gradient $\nabla\mathcal{L}$, respectively. From Eq. 5.10 it is clear that expression $\lambda_i \boldsymbol{u}_i \boldsymbol{v}_i^T$ can select useful signal in $\beta_j \boldsymbol{u}'_j \boldsymbol{v}'^T_j$ while separating the noise in $\epsilon_j \boldsymbol{u}^*_j \boldsymbol{v}^{*T}_j$ as long as their principal eigenvectors do not overlap fully. This idea is consistent with the PCA in which the largest and smallest eigenvalues of the scatter matrix are assumed to correspond to the signal and noise, respectively.

Complexity of Modulating Patterns. To further illustrate the modulation property of our approach, assume $\boldsymbol{M} \in \{0,1\}^{n_1 \times n_2}$ rather than just $\boldsymbol{M} \geq 0$. Furthermore, let $\boldsymbol{u}_i \in \{0,1\}^{n_1}$ and $\boldsymbol{v}_i \in \{0,1\}^{n_2}$. Then, expression $\vee_{i=1}^{r} \boldsymbol{u}_i \boldsymbol{v}_i^T$ becomes a binary modulator (switch or selector) for which r controls the complexity of selection pattern and $\vee$ performs the element-wise logical 'or' operation on matrices. For such a constrained problem, we have:

$$1+(2^{n_1}-1)(2^{n_2}-1) = \text{patt}(\boldsymbol{u}_1 \boldsymbol{v}_1^T) \leq \text{patt}(\vee_{i=1}^{r} \boldsymbol{u}_i \boldsymbol{v}_i^T) \leq \text{patt}(\boldsymbol{M}) = 2^{n_1 n_2}, \tag{5.11}$$

where $\text{patt}(\cdot)$ is the total number of unique patterns that can be generated for a given input matrix. For instance, if $n_1 = n_2 = 4$, we can generate 226, 8776, 49432, 65536 unique patterns for $r = 1, \cdots, 4$. Therefore, r controls the pattern complexity of our modulator. While for $\boldsymbol{M} \geq 0$ and $\boldsymbol{u}_i \in \mathbb{R}^{n_1}$ and $\boldsymbol{v}_i \in \mathbb{R}^{n_2}$ the expression for the pattern complexity may be more elaborate, the general principle of controlling the complexity of the modulating pattern via r holds.

Low-pass Filtering Property. The rank and singular values of the Hadamard product of two matrices are known to be upper-bounded as follows:

$$\begin{aligned} &\text{rank}(\boldsymbol{M} \odot \nabla\mathcal{L}) \leq \text{rank}(\boldsymbol{M}) \cdot \text{rank}(\nabla\mathcal{L}), \\ &\lambda_{max}(\boldsymbol{M} \odot \nabla\mathcal{L}) \leq \lambda_{max}(\boldsymbol{M}) \cdot \lambda_{max}(\nabla\mathcal{L}). \end{aligned} \tag{5.12}$$

These popular bounds do only tell that $\text{rank}(\boldsymbol{M} \odot \nabla\mathcal{L})$ could be lower than $\text{rank}(\nabla\mathcal{L})$ (the same holds for the largest singular values). However, by just simply looking at the rank-1 factorisation $\boldsymbol{M} = \boldsymbol{u}_1 \boldsymbol{v}_1^T$, we obtain:

$$(\boldsymbol{u}_1 \boldsymbol{v}_1^T) \odot \nabla\mathcal{L} = \text{diag}(\boldsymbol{u}_1) \cdot \nabla\mathcal{L} \cdot \text{diag}(\boldsymbol{v}_1), \tag{5.13}$$

where $\text{diag}(\cdot)$ is a diagonal matrix created from a vector. Thus, coefficients of $\boldsymbol{u}_1$ and $\boldsymbol{v}_1$ scale rows and columns of $\nabla\mathcal{L}$, and they can nullify entire rows and/or columns thus effectively decrease the rank so that $\text{rank}(\boldsymbol{M} \odot \nabla\mathcal{L}) < \text{rank}(\boldsymbol{M}) \cdot \text{rank}(\nabla\mathcal{L})$. Specifically, let $\nabla\mathcal{L}$ have r' singular values $\gamma_1 \leq \gamma_2 \leq \cdots \leq \gamma_{r'}$ and $\text{diag}(\boldsymbol{u}_1) \cdot \nabla\mathcal{L} \cdot \text{diag}(\boldsymbol{v}_1)$ have $k = r' - 1$ singular values $\gamma^*_1 \leq \gamma^*_1 \leq \cdots \leq \gamma^*_k$ due to the row and/or column deletion. Then, by the Cauchy's interlacing theorem, we have:

$$0 = \gamma^*_0 \leq \gamma_1 \leq \gamma^*_1 \leq \gamma_2 \leq \gamma^*_2 \leq \cdots \leq \gamma^*_k \leq \gamma_{r'}. \tag{5.14}$$

Now, $p_A = [\gamma_1, \cdots, \gamma_{r'}] / \sum_i \gamma_i$ and $p_B = [\gamma^*_1, \cdots, \gamma^*_k] / \sum_i \gamma^*_i$ can be thought of as two probability mass functions. As the support of p_B equals the support of p_A minus one, that is $\sup(p_B) = \sup(p_A) - 1$, and $\gamma^*_{i-1} \leq \gamma_i$ for $i = 1, \cdots, r'$, we have the following relation between the means and variances of both PMFs: (i) $\mu(p_B) < \mu(p_A)$ and (ii) $\sigma^2(p_B) < \sigma^2(p_A)$.

Due to points (i) and (ii), the Hadamard-based modulator $\boldsymbol{M} \odot \nabla\mathcal{L}$ can act as a low-pass filter.

Moreover, as our generic modulator goes over $i = 1, \cdots, r$, we have:

$$\boldsymbol{M} \odot \nabla\mathcal{L} = \sum_{i=0}^{r} \text{diag}(\boldsymbol{u}_i) \cdot \nabla\mathcal{L} \cdot \text{diag}(\boldsymbol{v}_i). \tag{5.15}$$

The higher the rank r of $\boldsymbol{M}$, the higher the rank of the entire modulated gradient can be. Specifically, we have:

$$\text{rank}(\boldsymbol{M}(\Psi, 1) \odot \nabla\mathcal{L}) \leq \text{rank}(\boldsymbol{M}(\Psi, 2) \odot \nabla\mathcal{L}) \leq \cdots \leq \text{rank}(\boldsymbol{M}(\Psi, r) \odot \nabla\mathcal{L}), \tag{5.16}$$

because when we iterate over i in Eq. 5.15 and aggregate, the consecutive contributions $\text{diag}(\boldsymbol{u}_i) \cdot \nabla\mathcal{L} \cdot \text{diag}(\boldsymbol{v}_i)$ can either fully/partially/not-at-all lie in the span of $\sum_{i'=1}^{i-1} \text{diag}(\boldsymbol{u}_{i'}) \cdot \mathcal{L} \cdot \text{diag}(\boldsymbol{v}_{i'})$.

Consider $\text{diag}(\boldsymbol{u}_i) \cdot \nabla\mathcal{L} \cdot \text{diag}(\boldsymbol{v}_i)$ and their corresponding PMFs p_{B_i} for $i = 1, \cdots, r$. Assume that each $\boldsymbol{u}_i \boldsymbol{v}_i^T$ is arranged as previously to have the ability to remove some row and/or column of $\nabla\mathcal{L}$. Based on Eq. 5.16 and Eq. 5.14, observe that combining random variables B_i by aggregation $\sum_{i=1}^{r} B_i$ (as Eq. 5.15 dictates) yields the following inequality which holds with a high probability if $\boldsymbol{u}_i \perp \boldsymbol{u}_j$ and $\boldsymbol{v}_i \perp \boldsymbol{v}_j$ for $i \neq j$:

$$\mu(p_{B_1}) \leq \mu(p_{B_1+B_2}) \leq \cdots \leq \mu(p_{B_1+B_2+\cdots+B_r}) \tag{5.17}$$

and

$$\sigma^2(p_{B_1}) \leq \sigma^2(p_{B_1+B_2}) \leq \cdots \leq \sigma^2(p_{B_1+B_2+\cdots+B_r}). \tag{5.18}$$

Synthetic experiment. Figure 5.9 presents a synthetic experiment with the following setting. We generate $\nabla\mathcal{L} \in \mathbb{R}^{10\times 10}$ from the uniform distribution, that is $\nabla\mathcal{L}_{mn} \sim \mathcal{U}(0,1)$. Moreover, we generate the modulation matrix $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{V}^T$ with $\boldsymbol{U} \in \mathbb{R}^{10\times r}$ and $\boldsymbol{V} \in \mathbb{R}^{10\times r}$ being drawn from the normal distribution *e.g.*, $U_{mn} \sim \mathcal{N}(0,1)$ and $V_{mn} \sim \mathcal{N}(0,1)$. Importantly, we note that the choice of the uniform and normal distributions can be swapped in an arbitrary way but it does not affect the conclusions of the following experiment.

Figure 5.9(a) shows that the lower the rank r is, the quicker the probability mass function (the trace normalized spectrum) decays. This illustrates that $\boldsymbol{M}(\cdot, r) \odot \nabla\mathcal{L}$ acts in fact as a low-pass filter on the gradient matrix $\nabla\mathcal{L}$. As the smallest singular values corresponding to the largest i represent the noise (by analogy to PCA), low-pass filtering helps us filter out the gradient noise.

Figures 5.9(b) and 5.9(c) show the corresponding mean and variance w.r.t. the rank r. Our ModGrad acts as a low-pass filter on the spectrum because the mean of the probability mass function for rank $r \leq 3$ is below the mean of the original PMF of $\nabla\mathcal{L}$. Similarly, the variance of the ModGrad-modulated spectrum is below the original variance for $\nabla\mathcal{L}$. These two figures validate the theoretical analysis/assertions of Eq. 5.17 and 5.18.

Figure 5.9(d) analyses the impact of sparsity of $\boldsymbol{U}$ and $\boldsymbol{V}$ on low-pass filtering. Specifically,

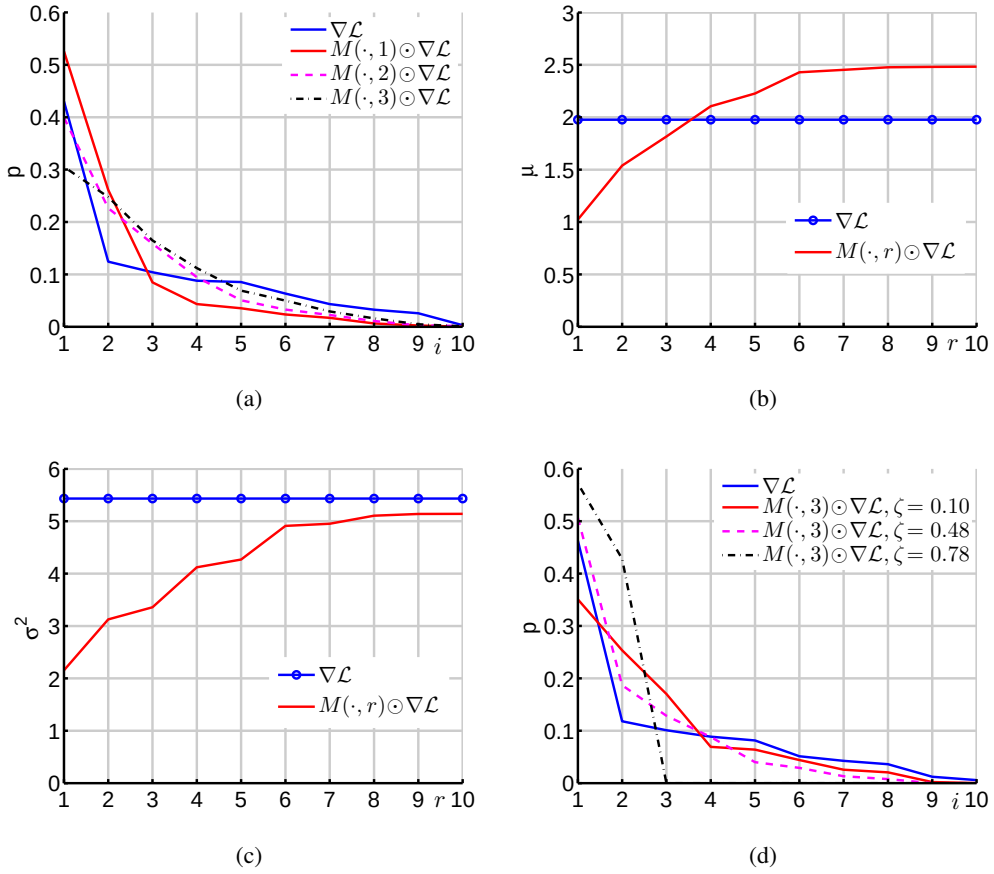
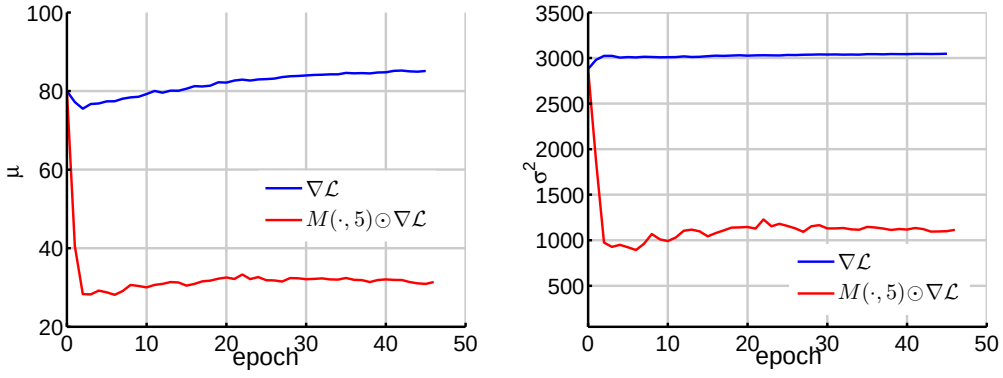


Figure 5.9: Synthetic experiments on the Hadamard-based modulator (see Section 5.5 for details). Figure 5.9(a) shows the probability mass function (the SVD singular values γ_i normalized by the trace and connected by line segments) for $\nabla\mathcal{L}$ and ModGrad of rank $r=1,2,3$, respectively. Figures 5.9(b) and Figure 5.9(c) show the corresponding mean and variance w.r.t. the rank r. Figure 5.9(d) shows the probability mass function for ModGrad of rank $r=3$ w.r.t. the sparsity level ζ.

we nullify entries U_{mn} and V_{mn} with the values below threshold τ^* which yields sparsity ζ (the number of nullified entries divided by the total number of coefficients of $\boldsymbol{U}$ and $\boldsymbol{V}$). Figure 5.9(d) shows that for $r=3$ the sparsity has a strong impact of lowering the rank of ModGrad. This finding is again consistent with the analysis in Eq. 5.14 regarding the ability of row and/or column scaling to lower the rank of $\boldsymbol{M}\odot\nabla\mathcal{L}$.

Real-data experiment. Figure 5.10 presents the results with and w/o ModGrad on *mini*-ImageNet. We reshape the gradient into matrices and we average them over the entire epoch. Similarly, we average the ModGrad-modulated matrices. We compute the SVD for each variant to obtain the singular values and we trace-normalize them to obtain the probability mass functions.

Figure 5.10(a) analyses the mean for the basic $\nabla\mathcal{L}$ *vs.* $\boldsymbol{M}\odot\nabla\mathcal{L}$. It is clear from the plot that our approach immediately reduces the mean of PMF and it keeps adjusting it over epochs for the best performance (see Table 5 in the main submission for the accuracy *vs.* the rank



(a) (b)

Figure 5.10: Experiments on the Hadamard-based modulator (see Section 5.5 for details) given the *mini*-ImageNet dataset. We analyze the probability mass function (the SVD singular values normalized by the trace and connected by line segments) for $\nabla\mathcal{L}$ and ModGrad of rank $r=5$. Figures 5.10(a) and 5.10(b) show the mean and variance w.r.t. the epoch number.

number). In contrast, the mean of PMF without ModGrad remains high throughout epochs.

Figure 5.10(b) analyses the variance for the basic $\nabla\mathcal{L}$ *vs.* $\boldsymbol{M} \odot \nabla\mathcal{L}$. Again, our GradMod immediately reduces the variance compared to the variance of PMF without ModGrad which remains high.

We note that while we analyse theoretically the ability of ModGrad to act as a low-pass spectral filter based on the Hadamard product, our design also benefits from the generator which produces matrix $\boldsymbol{M}$. The generator is exposed implicitly to gradients during the minimization of the loss function, therefore it is able to implicitly observe the directions directions helpful in minimizing the objective. The generative approach paired together with the adaptable low-pass filtering appear a robust combination to deal with noisy gradients.

Spectrum of row-wise scaled matrix $\boldsymbol{X}$. Analyzing the spectrum of row- and/or column-wise scaling is a somewhat complex matter but it can shed some light on the scaling properties of (the Hadamard product. Below we consider the toy case performing the row-wise scaling $\mathrm{diag}(\boldsymbol{u})\cdot\boldsymbol{X}$ where $\boldsymbol{u}\in\mathbb{R}^2_+$, $\mathrm{diag}(\boldsymbol{u})\in\mathcal{S}^2_+$ and $\boldsymbol{X}\in\mathcal{S}^2_{++}$. In what follows, we assume that eigenvalues we deal with are simple, that is $\lambda_1\neq\lambda_2$. Then, from the following system of equations

$$\begin{cases}\prod_{i=1}^2\lambda_i = \det(\mathrm{diag}(\boldsymbol{u})\cdot\boldsymbol{X}) = \det(\boldsymbol{X})\prod_{i=1}^2 u_i,\\ \sum_{i=1}^2\lambda_i = \mathrm{tr}(\mathrm{diag}(\boldsymbol{u})\cdot\boldsymbol{X}) = \sum_{i=1}^2 u_i\cdot X_{ii}\end{cases} \tag{5.19}$$

we obtain readily the following quadratic equation:

$$\lambda^2 - \mathrm{tr}(\mathrm{diag}(\boldsymbol{u})\cdot\boldsymbol{X})\cdot\lambda + \det(\boldsymbol{X})\prod_{i=1}^2 u_i = 0. \tag{5.20}$$

Solving Eq. 5.20 readily yields the following two eigenvalues for the row-scaled matrix $\boldsymbol{X}$:

$$\begin{cases}\lambda_1^u = \frac{1}{2}\mathrm{tr}(\mathrm{diag}(\boldsymbol{u})\cdot\boldsymbol{X}) + \frac{1}{2}\sqrt{\mathrm{tr}(\mathrm{diag}(\boldsymbol{u})\cdot\boldsymbol{X})^2 - 4\cdot\det(\boldsymbol{X})\prod_{i=1}^2 u_i},\\ \lambda_2^u = \frac{1}{2}\mathrm{tr}(\mathrm{diag}(\boldsymbol{u})\cdot\boldsymbol{X}) - \frac{1}{2}\sqrt{\mathrm{tr}(\mathrm{diag}(\boldsymbol{u})\cdot\boldsymbol{X})^2 - 4\cdot\det(\boldsymbol{X})\prod_{i=1}^2 u_i}.\end{cases} \tag{5.21}$$

From Eq. 5.21, we readily note that (i) $\lambda_2^u = 0$ if $u_i = 0$ for any $i \in \{1,2\}$, and (ii) as u_i is being gradually increased (step increases of u_i from 0 to ∞ for the chosen $i \in \{1,2\}$) and $\boldsymbol{u} > 0$, λ_1^u in Eq. 5.21 grows faster than λ_2^u for the majority of these step increases as the second part of both expressions (containing the square root) in Eq. 5.21 is added to/subtracted from $\frac{1}{2}\text{tr}(\text{diag}(\boldsymbol{u}) \cdot \boldsymbol{X})$ for λ_1^u and λ_2^u, respectively. This can be clearly seen as we compute:

$$\lim_{u_i \to \infty} \frac{\lambda_1^u}{\lambda_2^u} = \infty \cdot \text{sign}(u_j \cdot \det(\boldsymbol{X})), \; i \neq j, \tag{5.22}$$

where $\text{sign}(\cdot)$ returns the sign of the input expression.

> As the ratio $\lambda_1^u / \lambda_2^u \to \infty$ in Eq. 5.22 for $u_i \to \infty$ if $u_j > 0$ and $\det(\boldsymbol{X}) > 0$, the mean and variance of the underlying PMF decrease down to 1 and 0, respectively. Thus, Eq. 5.22 shows that setting a sufficiently large row-wise multiplication coefficient u_i, $i \in \{1,2\}$ for $\boldsymbol{X}$ makes the Hadamard product act as a low-pass filter.

The same observations hold for the singular values of the rectangular matrix $\boldsymbol{X} \in \mathbb{R}^{2 \times n_2}$, $n_2 \geq 2$ as Eq. 5.21 (*i.e.*, , the bottom equation) can be written as:

$$\lambda_2^u = \frac{1}{2}\text{tr}(\boldsymbol{\lambda}(\text{diag}(\boldsymbol{u}) \cdot \boldsymbol{X})) - \frac{1}{2}\sqrt{\text{tr}(\boldsymbol{\lambda}(\text{diag}(\boldsymbol{u}) \cdot \boldsymbol{X}))^2 - 4 \cdot \det(\boldsymbol{X}\boldsymbol{X}^T)^{\frac{1}{2}} \prod_{i=1}^{2} u_i}, \tag{5.23}$$

where $\boldsymbol{\lambda}(\cdot)$ is a matrix of singular values.

5.6 Chapter Conclusion

This chapter presents a meta-learner by modulating the gradient via so-called ModGrad. Our approach shows a general ability to address a wide range of problems including few-shot classification, regression and reinforcement learning. Empirical results show that ModGrad is competitive compared with other existing gradient-based meta-learners. Furthermore, ModGrad is designed to be modular (applicable to every layer) in deep neural networks. Thus, it can be utilized for other interesting applications without any extra structural changes to the base network architecture. In practice, our approach copes with the practical optimization matters such as learning rates, gradient step and adaptation to noise. Our gradient-based meta-learning algorithm remains robust in the presence of corrupted gradients while other existing methods have a low tolerance. Another benefit of ModGrad is the accelerated learning of the base network. As a result, ModGrad works also well with deeper networks, higher number of shots and a lower number of steps compared to MAML.

Chapter 6

On Learning the Geodesic Path for Incremental Learning

In this chapter, we aim to perform sequential adaptation for DNNs in class-incremental learning. The limitation to learn in this setting is the size of memory, in fact the storage cannot infinitely accumulate the samples. In this condition, the learning becomes challenging because some old samples might not be replayed in the future, and the model must be capable of adapting to new tasks without forgetting previously learned tasks. To this end, we propose a knowledge distillation method to prevent *catastrophic forgetting*. The idea is to introduce smooth properties between two tasks, and the connection is constructed using the geodesic flow. Our experiments show that the knowledge distillation considering the smooth properties can top the performance of previous knowledge distillation methods in class-incremental learning.

6.1 Introduction

Humans are able to gather and grow their knowledge gradually and effortlessly while preserving their *past knowledge*. In pursuing a similar capability at an algorithmic level and in the so-called Incremental Learning (IL), an artificial machine must learn and remember various tasks or concepts [Legg and Hutter, 2007] to accomplish a more broadened set of objectives. Formally, a learning algorithm in IL will receive its objectives (*e.g.*, recognizing new classes) gradually. This is in contrast to the conventional learning paradigm where the objectives and data are available from the beginning to the learning algorithm. The gradual nature of learning in IL poses serious difficulties as now the model has to sequentially learn and adapt to new tasks, while being vigilant to its needs and requirements (*e.g.*, limited memory to preserve the knowledge). As shown in the seminal works of [Kirkpatrick et al., 2017] and [Rebuffi et al., 2017b], the gradual nature of learning plus the presence of constraints and limitations in IL can degrade the performance of the model drastically, which is formally known as *"Catastrophic forgetting"* [McCloskey and Cohen, 1989; McClelland et al., 1995; French, 1999]. The phenomenon refers to a neural network experiencing performance degradation at previously learned concepts when trained sequentially on learning new concepts. This forgetting problem appears even worse when the model is sequentially updated for new tasks without considering the previous tasks as shown in [Kirkpatrick et al., 2017; Li and Hoiem, 2017] *i.e.*, learning new tasks overrides the knowledge from previous tasks. Finding the balance among tasks, also

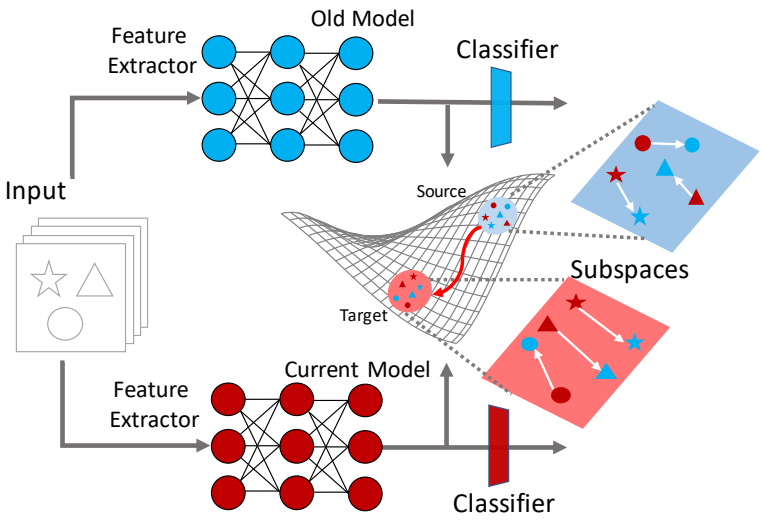


Figure 6.1: The visualization of our proposed method. To regularize the network for incoming tasks, knowledge from previous tasks is preserved by distilling the features following the geodesic path between two subspaces of different models. The distillation is based on the projection of two sets of features from two different networks.

known as the stability-plasticity dilemma in [Rasch and Born, 2013], is crucial to achieve the IL ultimate goal.

In classical classification on visual data, a Convolutional Neural Network (CNN) is used to encode the input images into features and a final layer consists of classifiers to map the features to the fixed number of classes. In class IL [Van de Ven and Tolias, 2019], a CNN keeps learning and grows its capacity to accommodate new classes or tasks. In order to achieve the equilibrium performance between learning new tasks and maintaining existing base performance, some IL methods store observed tasks in the memory and replay them [Gepperth and Karaoguz, 2016; Lopez-Paz and Ranzato, 2017] to prevent *catastrophic forgetting*. However, learning through a large number of tasks limits the exemplars that can be reserved in the memory. Hou *et al*. [Hou et al., 2019b] suggest to replay the exemplars in the memory on both old and current models and prevent the forgetting phenomenon with knowledge distillation.

The design of distillation loss remains an open research problem. A knowledge distillation on the output space (after the classifier) is firstly proposed in [Li and Hoiem, 2017] so-called Learning without Forgetting (LwF). However, the study in [Rebuffi et al., 2017b; Hou et al., 2019b] shows that LwF lets the parameters of the new classes to become more dominant than the old parameters. As a result, the model tends to classify all test data to the new classes. Because of this reason, the classifier is created based on the *nearest mean of exemplars* and the distillation loss with the cosine distance in the feature space is suggested in [Hou et al., 2019b]. All these prior methods do not consider the gradual change between tasks. In fact, human minds learn from one task to another task by *gradual walk* as described in [Chalup, 2002]. Inspired by the idea of gradual change in human minds, we propose a distillation loss in IL by adopting the concept of geodesic flow between two tasks (see Fig. 6.1) called as **GeoDL**. To summarize, the contributions of our work are:

i. We propose a novel distillation loss that considers the information geometry aspect in

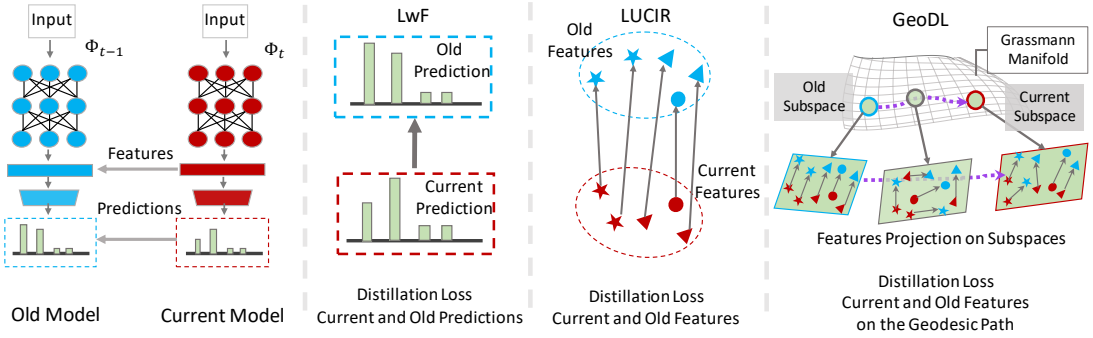


Figure 6.2: Comparison with prior knowledge distillation approaches for IL. Knowledge distillation is applied to the features or predictions from the old model $\mathbf{\Phi}_{t-1}$ and the current model $\mathbf{\Phi}_t$. LwF [Li and Hoiem, 2017] distillation loss is based on the predictions and LUCIR [Hou et al., 2019b] preserves the knowledge from previous tasks using a cosine embedding loss. Our proposal, called GeoDL, makes use of *gradual walk* between two subspaces from old and current features.

incremental learning and leads to improved performance.

ii. We show that using a geodesic path for knowledge distillation yields less forgetful networks compared to the distillation losses using the Euclidean metric.

6.2 Preliminaries

Evaluating an IL Model. There are two important objectives to assess the performance of a model in IL problems. **1)**. The average accuracy ($\mathcal{A}$) to evaluate the capability of the model in learning new tasks. **2)**. Forgetting rate ($\mathcal{F}$) to evaluate how much the model suffers from *catastrophic forgetting*. For each task, we compute an accuracy score $\mathcal{A}_t$ for a test set that is representative for all classes seen so far (*i.e.*, classes seen from $\mathcal{D}_0$ up to $\mathcal{D}_t$). We quantify the forgetting rate by considering the accuracy for the test set at $t = 0$ using the model $\mathbf{\Phi}_0$ and the final model $\mathbf{\Phi}_T$. The average accuracy and forgetting rate are defined as:

$$\mathcal{A} = \frac{1}{T}\sum_{t=1}^{T}\mathcal{A}_t, \qquad \mathcal{F} = \mathcal{A}_0|_{\mathbf{\Phi}_0} - \mathcal{A}_0|_{\mathbf{\Phi}_T}\ . \tag{6.1}$$

Here, we overloaded our notations slightly and denote the accuracy of a model $\mathbf{\Phi}_t$ using the test set of the task at time $t = 0$ by $\mathcal{A}_0|_{\mathbf{\Phi}_t}$. A successful model is one that has a high average accuracy and low forgetting rate.

6.2.1 Regularization with knowledge distillation

We start this part by reviewing how prior works mitigate the forgetting phenomenon with knowledge distillation. We then discuss how our proposed method extends the idea of distillation loss by considering the gradual change between tasks (see Fig. 6.2 for a conceptual diagram that highlights the differences between our approach and prior art). To prevent *catastrophic*

forgetting, distillation loss aims to minimize alteration on shared network parameters during the adaptation process (*i.e.*, learning a novel task). The distillation loss is employed along with the cross-entropy loss [Rebuffi et al., 2017b; Hou et al., 2019b; Liu et al., 2020b] or the triplet loss [Yu et al., 2020]. In the Learning without Forget (LwF) [Li and Hoiem, 2017], the knowledge distillation is applied to minimize the dissimilarity between predictions of the old and current models. This is to ensure that predictions on previously seen classes do not vary drastically (*i.e.*, $p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\Phi}_t) \approx p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\Phi}_{t-1})$), leading to maintaining prior knowledge. Formally and for a problem with K classes, the loss of LwF is defined as:

$$\begin{aligned}\mathcal{L}_{\text{LwF}}(\boldsymbol{\Phi}_t, \boldsymbol{\Phi}_{t-1}) &= \sum_{k=1}^{K} \big[p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\Phi}_{t-1})\big]_k \log \big[p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\Phi}_t)\big]_k \\ \big[p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\Phi}_t)\big]_k &= \frac{\exp(f(\boldsymbol{x}, \boldsymbol{\Phi}_t)_k/\tau)}{\sum_{k=1}^{K} \exp(f(\boldsymbol{x}, \boldsymbol{\Phi}_t)_k/\tau)} \,. \end{aligned} \tag{6.2}$$

Here, $[\cdot]_k$ denotes the element at index k of a vector and τ is a temperature variable. Note that $\boldsymbol{y}$ belongs to the old class in $\mathcal{D}_{t-1}$.

Another form of distillation , again aiming to minimize the dissimilarity between predictions, is to constraint the latent features of the network (*i.e.*, $\boldsymbol{z} = f(\boldsymbol{x}; \boldsymbol{\theta})$). Hou et al. [2019b] propose the so-called *less-forget* constraint which employs a cosine embedding loss for knowledge distillation. Formally, we have:

$$\mathcal{L}_{\text{Cos}}(\boldsymbol{\theta}_t, \boldsymbol{\theta}_{t-1}) = \sum_{\boldsymbol{x} \sim \{\mathcal{D}_t \cup \mathcal{M}\}} \big[1 - \text{sim}\big(f(\boldsymbol{x}; \boldsymbol{\theta}_t), f(\boldsymbol{x}; \boldsymbol{\theta}_{t-1})\big)\big],$$

where:

$$\text{sim}(\boldsymbol{u}, \boldsymbol{v}) = \frac{\boldsymbol{u}^\top \boldsymbol{v}}{\|\boldsymbol{u}\| \|\boldsymbol{v}\|}. \tag{6.3}$$

The cosine distance used in Eq. 6.3 is based on the Euclidean metric between features from two models directly. The aforementioned distillation losses do not benefit from the manifold structure of each task.

6.3 Proposed Method

As described in §6.2.1, the existing methods for knowledge distillation do not take into account the gradual change between consecutive tasks. Furthermore, the Euclidean norm often used for distillation cannot capture the underlying geometry of two different feature spaces. Furthermore, the previous works did not explicitly benefit from the geometry and manifold structure of the tasks during performing distillation. To benefit from the manifold geometry during distillation, we propose to enforce consistency along the geodesic connecting the models $\boldsymbol{\Phi}_{t-1}$ and $\boldsymbol{\Phi}_t$ (see Fig. 6.3 for a conceptual diagram).

6.3.1 Gradual walk with intermediate subspaces

We use the concept of subspaces to embed a set of features (*e.g.*, a mini-batch) from both old and current models. To this end, we model the geometry of latent samples in a model by a

low-dimensional subspace with a basis[1] $\boldsymbol{P} \in \mathbb{R}^{\mathrm{d}\times\mathrm{n}}$. Subspaces form a Riemannian manifold are studied using the geometry of Grassmann manifolds.

Definition 2 (Grassmann Manifold)**.** *The set of n-dimensional linear subspaces of* $\mathbb{R}^d, 0 < n < d$ *("linear" will be omitted in the sequel) is termed the Grassmann manifold, and is denoted here by* $\mathcal{G}(\mathrm{n},\mathrm{d})$*. An element* $\boldsymbol{P}$ *of* $\mathcal{G}(\mathrm{n},\mathrm{d})$ *can be specified by a basis,* i.e.*, a* $d \times n$ *matrix with orthonormal columns (*i.e.*,* $\mathcal{G}(\mathrm{n},\mathrm{d}) \ni \boldsymbol{P} \Rightarrow \boldsymbol{P}^\top \boldsymbol{P} = \mathbf{I}_n$*).*

The *gradual walk* is essentially the geodesic flow between two points on the Grassmann manifold. In contrast to the distillation losses using the Euclidean metric, our approach allows the features to be projected along the geodesic flow and can capture the smooth changes between tasks.

Given a batch of B data samples, let $\boldsymbol{Z}_{t-1}, \boldsymbol{Z}_t \in \mathbb{R}^{\mathrm{d}\times B}$ be the corresponding encodings by the model at time $t-1$ and t, respectively. We propose to model $\boldsymbol{Z}_{t-1}$ and $\boldsymbol{Z}_t$ by two low-dimensional manifolds. This, in our work is indeed the two subspaces spanning $\boldsymbol{Z}_{t-1}$ and $\boldsymbol{Z}_t$ which we denote by $\boldsymbol{P}_{t-1}$ and $\boldsymbol{P}_t$. The geodesic flow between $\boldsymbol{P}_{t-1}$ and $\boldsymbol{P}_t$ denoted by $\boldsymbol{\Pi} : \nu \in [0,1] \rightarrow \boldsymbol{\Pi}(\nu) \in \mathcal{G}(\mathrm{n},\mathrm{d})$ is:

$$\boldsymbol{\Pi}(\nu) = \begin{bmatrix} \boldsymbol{P}_{t-1} & \boldsymbol{R} \end{bmatrix} \begin{bmatrix} \boldsymbol{U}_1 \boldsymbol{\Gamma}(\nu) \\ -\boldsymbol{U}_2 \boldsymbol{\Sigma}(\nu) \end{bmatrix}, \tag{6.4}$$

where $\boldsymbol{R} \in \mathbb{R}^{\mathrm{d}\times(\mathrm{d}-\mathrm{n})}$ is the orthogonal complement of $\boldsymbol{P}_{t-1}$ and the diagonal elements for $\boldsymbol{\Gamma}(\nu)$ and $\boldsymbol{\Sigma}(\nu)$ are $\gamma_i = \cos(\nu\omega_i)$ and $\sigma_i = \sin(\nu\omega_i)$, respectively for $i = 1, 2, \cdots, \mathrm{n}$ and $0 \leq \omega_1 \leq \cdots \leq \omega_\mathrm{n} \leq \pi/2$. In the supplementary material of our paper, we provide more details about the geodesic flow on Grassmannian and how $\boldsymbol{R}$, $\boldsymbol{\Gamma}(\nu)$ and $\boldsymbol{\Sigma}(\nu)$ can be obtained from $\boldsymbol{P}_{t-1}$ and $\boldsymbol{P}_t$.

Note that both $\boldsymbol{P}_{t-1}^\top \boldsymbol{P}_t$ and $\boldsymbol{R}^\top \boldsymbol{P}_t$ share the same right singular vectors $\boldsymbol{V}$, thus generalized Singular Value Decomposition (SVD) [Van Loan, 1976] can be employed to decompose the matrices.

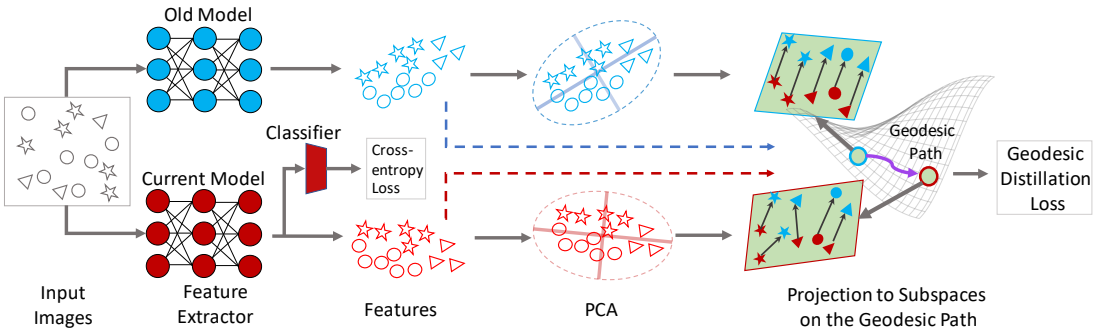


Figure 6.3: Pipeline of our proposed approach. We model the feature space of the previous $\boldsymbol{\Theta}_{t-1}$ and the current task $\boldsymbol{\Theta}_t$ by two low-dimensional subspaces and enforce distillation along the geodesic connecting them. This is achieved by aligning samples along projection on the subspaces on the geodesic flow.

[1] Principal Component Analysis (PCA) can be used to obtain the basis of the subspace.

6.3.2 Projection to the intermediate subspaces

We propose to distill the feature space of the previous model into the current one by considering the cosine similarity between projections along $\mathbf{\Pi}(\nu)$. Let $z_{t-1} = f(\boldsymbol{x}, \mathbf{\Phi}_{t-1})$ and $z_t = f(\boldsymbol{x}, \mathbf{\Phi}_t)$ be the encodings of $\boldsymbol{x}$ according to the previous and current models, respectively. The inner product between projections of z_{t-1} and z_t on a subspace $\boldsymbol{P}$ is:

$$\begin{aligned} \mathrm{g}_{\boldsymbol{P}}(z_{t-1}, z_t) &= \left\langle \boldsymbol{P}^\top z_{t-1}, \boldsymbol{P}^\top z_t \right\rangle = \left(\boldsymbol{P}^\top z_{t-1}\right)^\top \boldsymbol{P}^\top z_t \\ &= \boxed{z_{t-1}^\top \boldsymbol{P}\boldsymbol{P}^\top z_t}. \end{aligned}$$

From this, we can define the inner product along the geodesic flow as:

$$\begin{aligned} \mathrm{g}_{\Pi}(z_{t-1}, z_t) &\coloneqq \int_0^1 \mathrm{g}_{\Pi(\nu)}(z_{t-1}, z_t)\, d\nu \\ &= \int_0^1 z_{t-1}^\top \mathbf{\Pi}(\nu)\mathbf{\Pi}(\nu)^\top z_t \, d\nu \\ &= z_{t-1}^\top \underbrace{\left(\int_0^1 \mathbf{\Pi}(\nu)\mathbf{\Pi}(\nu)^\top \, d\nu \right)}_{Q} z_t. \end{aligned}$$

Let $Q = \mathbf{\Delta}\mathbf{\Lambda}\mathbf{\Delta}^\top$. It can be shown that:

$$\begin{aligned} \mathbf{\Delta} &= \begin{bmatrix} \boldsymbol{P}_{t-1}\boldsymbol{U}_1 & \boldsymbol{R}\,\boldsymbol{U}_2 \end{bmatrix}, \\ \mathbf{\Lambda} &= \int_0^1 \begin{bmatrix} \mathbf{\Gamma}(\nu)\mathbf{\Gamma}(\nu) & -\mathbf{\Gamma}(\nu)\mathbf{\Sigma}(\nu) \\ -\mathbf{\Sigma}(\nu)\mathbf{\Gamma}(\nu) & \mathbf{\Sigma}(\nu)\mathbf{\Sigma}(\nu) \end{bmatrix} d\nu. \end{aligned} \tag{6.5}$$

Recall $\gamma_i = \cos(\omega_i)$ and $\sigma_i = \sin(\omega_i)$, we can calculate the diagonal elements:

$$\begin{aligned} \lambda_{1i} &= \int_0^1 \cos^2(\nu\omega)\, d\nu = 1 + \frac{\sin(2\omega_i)}{2\omega_i}, \\ \lambda_{2i} &= -\int_0^1 \cos(\nu\omega)\sin(\nu\omega)\, d\nu = \frac{\cos(2\omega_i) - 1}{2\omega_i}, \\ \lambda_{3i} &= \int_0^1 \sin^2(\nu\omega)\, d\nu = 1 - \frac{\sin(2\omega_i)}{2\omega_i}. \end{aligned} \tag{6.6}$$

This will enable us to compute Q in closed-form as (see our supplementary material for the detailed derivations):

$$Q = \mathbf{\Delta} \begin{bmatrix} \lambda_1 & \lambda_2 \\ \lambda_2 & \lambda_3 \end{bmatrix} \mathbf{\Delta}^\top, \tag{6.7}$$

where Q is a $\mathrm{d} \times \mathrm{d}$ positive semi-definite matrix. In the final form, the inner product between features projected onto intermediate subspaces is:

$$\mathrm{g}_{\Pi}(z_{t-1}, z_t) = z_t^\top Q z_{t-1}. \tag{6.8}$$

Intuitively, $\boldsymbol{Q}$ is the matrix that defines the manifold structure between features of two different tasks. We argue that this facilitates distilling the knowledge by taking into account the smooth changes between the low-dimensional models (*i.e.*, manifolds) of the tasks. To use g_{Π} for training, we propose to minimize the following distillation loss, which can be understood as a general form of cosine similarity when the geodesic flow is considered:

$$\mathcal{L}_{\text{GeoDL}} = 1 - \frac{\boldsymbol{z}_t^\top \boldsymbol{Q} \boldsymbol{z}_{t-1}}{\|\boldsymbol{Q}^{1/2}\boldsymbol{z}_t\| \|\boldsymbol{Q}^{1/2}\boldsymbol{z}_{t-1}\|}. \tag{6.9}$$

In practice, in order to better prevent *catastrophic forgetting*, we use an adaptive weighting scheme for GeoDL as $\beta_{ad} = \beta\sqrt{|N_{new}|/|N_{old}|}$. Note that this weighting scheme is a common practice for recent IL algorithms [Hou et al., 2019b; Liu et al., 2020b]. Interestingly, one can recover the conventional distillation loss in Eq. 6.3 by choosing $\boldsymbol{P}_{t-1} = \boldsymbol{P}_t$, resulting in $\boldsymbol{Q} = \mathbf{I}$. This reveals that the conventional distillation loss is indeed a special case of our solution.

Algorithm 5 Train IL with GeoDL

Input: $\mathcal{D}_0, \cdots, \mathcal{D}_T$

$\boldsymbol{\Phi}_0 \leftarrow$ random initialization
Train $\boldsymbol{\Phi}_0$ on $\mathcal{D}_0$ minimizing $\mathcal{L}_{\text{CE}}$
Select and store exemplars to $\mathcal{M}$
for t in $\{1, ..., T\}$ **do**
    while not done **do**
        Sample a mini-batch $\{\boldsymbol{X}, \boldsymbol{Y}\}$ from $\{\mathcal{D}_t \cup \mathcal{M}\}$
        Get $\boldsymbol{Z}_{t-1} = f(\boldsymbol{X}, \boldsymbol{\Phi}_{t-1}), \boldsymbol{Z}_t = f(\boldsymbol{X}, \boldsymbol{\Phi}_t)$
        Compute $\boldsymbol{P}_{t-1} = \text{PCA}(\boldsymbol{Z}_{t-1})$
        Compute $\boldsymbol{P}_t = \text{PCA}(\boldsymbol{Z}_t)$
        Generate the geodesic flow using Eq. 6.7
        Project $\boldsymbol{Z}_t$ and $\boldsymbol{Z}_{t-1}$ to the geodesic path
        Minimize $\mathcal{L}_{\text{CE}}$ and $\mathcal{L}_{\text{GeoDL}}$
        Update $\boldsymbol{\Phi}_t$
    end while
    Evaluate on the test set
    Update exemplars from $\mathcal{D}_t$ and $\mathcal{M}$
end for

Below, we explain the details on how to generate the geodesic flow. Recall the subspaces from the old model $\boldsymbol{P}_{t-1}$, the current model $\boldsymbol{P}_t$, and the orthogonal complement $\boldsymbol{R}$ are used to compute the geodesic flow at ν:

$$\boldsymbol{\Pi}(\nu) = \begin{bmatrix} \boldsymbol{P}_{t-1} & \boldsymbol{R} \end{bmatrix} \begin{bmatrix} \boldsymbol{U}_1 \boldsymbol{\Gamma}(\nu) \\ -\boldsymbol{U}_2 \boldsymbol{\Sigma}(\nu) \end{bmatrix}. \tag{6.10}$$

We decompose $\boldsymbol{P}_{t-1}^\top \boldsymbol{P}_t$ and $\boldsymbol{R}^\top \boldsymbol{P}_t$ via the generalized SVD [Van Loan, 1976] to obtain the

orthonormal matrices $\boldsymbol{U}_1$ and $\boldsymbol{U}_2$:

$$\begin{aligned} \boldsymbol{P}_{t-1}^{\top}\boldsymbol{P}_t &= \boldsymbol{U}_1\boldsymbol{\Gamma}(1)\boldsymbol{V}^{\top}, \\ \boldsymbol{R}^{\top}\boldsymbol{P}_t &= -\boldsymbol{U}_2\boldsymbol{\Sigma}(1)\boldsymbol{V}^{\top}. \end{aligned} \tag{6.11}$$

All intermediate time steps $\nu \in (0,1)$ on the geodesic path are used for feature projection $\boldsymbol{\Pi}(\nu)^{\top}\boldsymbol{z}$ for obtaining the similarity in our distillation loss. We note that it is not necessary to compute or store all projection into the intermediate subspace because a closed form solution can be computed as follows:

$$\boldsymbol{Q} = \boldsymbol{\Delta}\begin{bmatrix}\lambda_1 & \lambda_2 \\ \lambda_2 & \lambda_3\end{bmatrix}\boldsymbol{\Delta}^{\top}, \tag{6.12}$$

where:

$$\boldsymbol{\Delta} = \begin{bmatrix}\boldsymbol{P}_{t-1}\boldsymbol{U}_1 & \boldsymbol{R}\,\boldsymbol{U}_2\end{bmatrix}, \tag{6.13}$$

$$\begin{aligned} \lambda_{1i} &= 1 + \frac{\sin(2\omega_i)}{2\omega_i}, \\ \lambda_{2i} &= \frac{\cos(2\omega_i) - 1}{2\omega_i}, \\ \lambda_{3i} &= 1 - \frac{\sin(2\omega_i)}{2\omega_i}. \end{aligned} \tag{6.14}$$

We can calculate $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2$, and $\boldsymbol{\lambda}_3$ by using diagonal elements of $\boldsymbol{\Gamma}(1)$ and calculating $\omega_i = \arccos(\gamma_i)$. Note that, the value of γ_i is clamped between -1 and 1 for computational stability.

Algorithm 6 provides details of how we generate the geodesic flow.

Algorithm 6 Generate the Geodesic Flow

Input: The subspaces of the old model $\boldsymbol{P}_{t-1}$ and the current model $\boldsymbol{P}_t$

1: Get the orthogonal complement $\boldsymbol{R}$ of $\boldsymbol{P}_{t-1}$
2: Compute $\boldsymbol{A} = \boldsymbol{P}_{t-1}^{\top}\boldsymbol{P}_t$ and $\boldsymbol{B} = \boldsymbol{R}^{\top}\boldsymbol{P}_t$
3: Decompose $\boldsymbol{A}, \boldsymbol{B}$ using gen. SVD to obtain $\boldsymbol{\Sigma}, \boldsymbol{\Gamma}, \boldsymbol{U}_1, \boldsymbol{U}_2$
4: Compute $\boldsymbol{\omega}$ from the diag. elements of $\boldsymbol{\Gamma}(1)$
5: Compute $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2$, and $\boldsymbol{\lambda}_3$ using Eq. 6.14.
6: Compute $\boldsymbol{Q}$ using the closed-form solution in Eq. 6.12
7: Return the generated geodesic flow $\boldsymbol{Q}$

6.3.3 Classifiers and exemplars selection

Below we discuss the classifier design and sample selection which are also important for IL. Both of these methods are widely known for tasks with visual data (*e.g.*, see [Hou et al., 2019b; Chen et al., 2019b; Rebuffi et al., 2017b; Mensink et al., 2012; Gidaris and Komodakis, 2018a]). Suppose the classifier encodes the template of class i by a normalized vector $\boldsymbol{\varphi}_i / \|\boldsymbol{\varphi}_i\|$. The

likelihood of class i can be obtained as:

$$p(y_i|\boldsymbol{x}) = \frac{\exp\big(\text{sim}(\boldsymbol{\varphi}_i, \boldsymbol{z}_t)\big)}{\sum_j \exp\big(\text{sim}(\boldsymbol{\varphi}_j, \boldsymbol{z}_t)\big)}. \tag{6.15}$$

Note that $\boldsymbol{\varphi}$ is updated by SGD along other network parameters. The classification uses the cross-entropy loss $\mathcal{L}_{\text{CE}}$.

In order to select exemplars to be stored in the memory, we use the *herding* method by selecting top-k samples with the nearest cosine distance to the mean-embedding classifier. Algorithm 5 details out the steps of our approach. Note that our approach is an end-to-end training algorithm, thus the algorithm learns embeddings which transition smoothly between tasks thanks to the geodesic flow.

6.4 Experiments

In this section, we examine our proposed method and compare with the state of the art on several datasets: CIFAR-100, ImageNet-subset, and ImageNet-full.

6.4.1 Dataset and implementation

The CIFAR-100 dataset [Krizhevsky, 2009] contains 100 classes and each class has 500 train and 100 test images with the size of 32×32. The ImageNet dataset is evaluated for two different splits: ImageNet-subset with 100 categories and ImageNet-full with 1000 categories. Both ImageNet and CIFAR-100 datasets are used to examine our method. Basic data augmentation is employed when training our algorithm *e.g.*, image flipping, color jitter, and random crop for all datasets. For a fair comparison, arrangement of all classes in our experiment closely follows a random seed (1993) in the same way as implemented in [Hou et al., 2019b; Liu et al., 2020b] and our performance is evaluated on three different runs. The first sequence ($t = 0$) is trained with 50 classes for CIFAR-100 and ImageNet-subset and 500 classes for ImageNet-full, afterwards the IL tasks *e.g.*, 5, 10, and 25 are performed to evaluate and compare our proposed approach with the state-of-the-art techniques.

The CNN backbone used for comparison on CIFAR-100 is ResNet-32 [He et al., 2016] following the network architecture in [Hou et al., 2019b]. On ImageNet-subset and ImageNet-full, we evaluate the proposed approach using ResNet-18 [He et al., 2016]. The Stochastic Gradient Descent (SGD) optimizer is applied to train the CNN for all experiments with an initial learning rate set to 0.1 and reduced by a factor 0.1 every half and three quarters of the total epochs. The model is trained on each sequence with 160, 90, and 90 epochs on CIFAR-100, ImageNet-subset, and ImageNet-full, respectively. We set the mini-batch size 128 and $\beta = 6.0$ for all datasets. We fix the memory size and set to 20 exemplars per class for all experiments unless otherwise specified. We use the PyTorch package [Paszke et al., 2017] with automatic differentiation in our implementation.

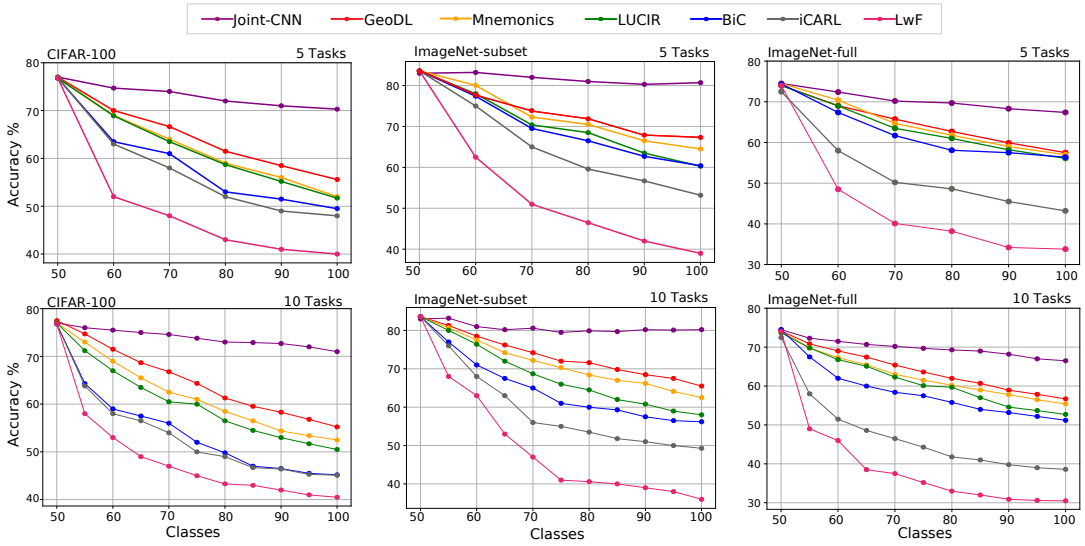


Figure 6.4: The accuracy for each task (5 and 10 tasks) on CIFAR-100 (left), ImageNet-subset (middle), and ImageNet-full (right). The x-axis shows the number of classes learned at a specific time and y-axis is the corresponding accuracy.

6.4.2 Evaluations

To evaluate our technique, we compare with the state of the art in IL, namely LwF [Li and Hoiem, 2017], iCARL [Rebuffi et al., 2017b], BiC [Wu et al., 2019], LUCIR [Hou et al., 2019b], and Mnemonics [Liu et al., 2020b]. Note that, we cite the results of Mnemonics [Liu et al., 2020b] from the corrected version on arXiv that is different from the published version (CVPR2020). Among all, LwF [Li and Hoiem, 2017] and LUCIR [Hou et al., 2019b] are the methods that propose knowledge distillation to prevent *catastrophic forgetting*. To provide reference, we also report the upper-bound (joint-CNN) in which all data from previous tasks is available and not limited to only samples from the memory. We adopt several baselines to compare with the-state-of-the-art methods. For selecting exemplars in the memory (*herding*), we employ the *nearest-mean-selection* technique and the *nearest-mean-of-exemplars* classifier in iCARL [Rebuffi et al., 2017b] with GeoDL. Furthermore, we also incorporate our method with LUCIR [Hou et al., 2019b] by replacing the feature distillation loss $\mathcal{L}_{\text{cos}}$ by our distillation loss $\mathcal{L}_{\text{GeoDL}}$ while the cross-entropy loss and the margin ranking loss remain in the objective function. We show and compare our method on CIFAR-100, ImageNet-subset, and ImageNet-full in Table 6.1. In addition, we also plot the accuracy of GeoDL + LUCIR in comparison with prior methods in Fig. 6.4. Please refer to our supplementary material for additional plots and experiments.

Results on CIFAR-100. Table 6.1 provides comparisons with the state-of-the-art methods on CIFAR-100. GeoDL improves the basic iCARL [Rebuffi et al., 2017b] method (without knowledge distillation) by 8%, 13%, and 15% for 5, 10, and 25 tasks, respectively. Furthermore, our GeoDL also improves the LUCIR algorithm [Hou et al., 2019b] by 2%, 5%, and 5% for 5,

Method	**CIFAR-100**			**ImageNet-subset**			**ImageNet-full**		
Average accuracy (%)	5	10	25	5	10	25	5	10	25
LwF [Li and Hoiem, 2017]	49.59	46.98	45.51	53.62	47.64	44.32	44.35	38.90	36.87
iCARL [Rebuffi et al., 2017b]	57.12	52.66	48.22	65.44	59.88	52.97	51.50	46.89	43.14
BiC [Wu et al., 2019]	59.36	54.20	50.00	70.07	64.96	57.73	62.65	58.72	53.47
LUCIR [Hou et al., 2019b]	63.17	60.14	57.54	70.84	68.32	61.44	64.45	61.57	56.56
Mnemonics [Liu et al., 2020b]	63.34	62.28	60.96	72.58	71.37	69.74	64.63	63.01	61.00
iCARL + GeoDL	62.54	61.40	61.84	70.10	70.86	70.72	60.02	57.98	56.70
LUCIR + GeoDL	**65.14**	**65.03**	**63.12**	**73.87**	**73.55**	**71.72**	**65.23**	**64.46**	**62.20**
Forgetting rate (%)	5	10	25	5	10	25	5	10	25
LwF [Li and Hoiem, 2017]	43.36	43.58	41.66	55.32	57.00	55.12	48.70	47.94	49.84
iCARL [Rebuffi et al., 2017b]	31.88	34.10	36.48	43.40	45.84	47.60	26.03	33.76	38.80
BiC [Wu et al., 2019]	31.42	32.50	34.60	27.04	31.04	37.88	25.06	28.34	33.17
LUCIR [Hou et al., 2019b]	18.70	21.34	26.46	31.88	33.48	35.40	24.08	27.29	30.30
Mnemonics [Liu et al., 2020b]	10.91	13.38	19.80	17.40	17.08	20.83	13.85	15.82	19.17
iCARL + GeoDL	12.20	21.10	26.84	26.84	22.44	24.88	21.84	22.87	28.22
LUCIR + GeoDL	**9.49**	**9.10**	**12.01**	**13.78**	**12.68**	**15.21**	**11.03**	**12.81**	**15.11**

Table 6.1: The average accuracy and the forgetting rate on ImageNet-subset. The numbers of tasks T are set to 5, 10, and 25. Ideally, each method must find the balance to achieve the high average accuracy and the low forgetting rate.

10, and 25 tasks, respectively. Our method also has lower forgetting rates in contrast to recent IL algorithms across different numbers of tasks. Fig. 6.4 (left) also shows that GeoDL consistently achieves higher accuracy than knowledge distillation based methods *e.g.*, LUCIR [Hou et al., 2019b] and LwF [Li and Hoiem, 2017] for each task. Our method tops the balance to learn new tasks and avoid *catastrophic forgetting*.

Results on ImageNet. Table 6.1 also reports the average accuracy of our model. Fig. 6.4 (middle) shows the accuracy for each task in comparison to prior methods for IL on ImageNet-subset. Our results suggest that GeoDL improves the baseline methods by notable margin. The basic iCARL [Rebuffi et al., 2017b] method without GeoDL achieves the classification accuracy 70.10%, 70.86%, and 70.72% for 5, 10, and 25 tasks, respectively. Furthermore, GeoDL also improves the LUCIR algorithm [Hou et al., 2019b] with the highest accuracy 73.87%, 73.55%, and 71.72% for 5, 10, and 25 tasks, respectively. Our method also has low forgetting rates across different numbers of tasks. The evaluation on ImageNet-full shows that our method outperforms the prior methods for IL where it achieves high average accuracy and prevents *catastrophic forgetting*.

6.4.3 Ablation studies

Below, we investigate how our method improves the basic approach for IL without considering additional loss functions and exemplars selection. We only use the cross-entropy loss and the *herding* selection method [Rebuffi et al., 2017b] for exemplars in our ablation studies. We compare GeoDL with the knowledge distillation approaches proposed in LwF [Li and Hoiem, 2017] and LUCIR [Hou et al., 2019b]. In this study, we examine these distillation losses and ours w.r.t. various classifiers and number of exemplars in the memory on CIFAR-100.

Impact of different classifiers. We employ three different classifiers to examine the behavior of our method. The first classifier is based on learnable class prototypes $\boldsymbol{\varphi}$, one per class.

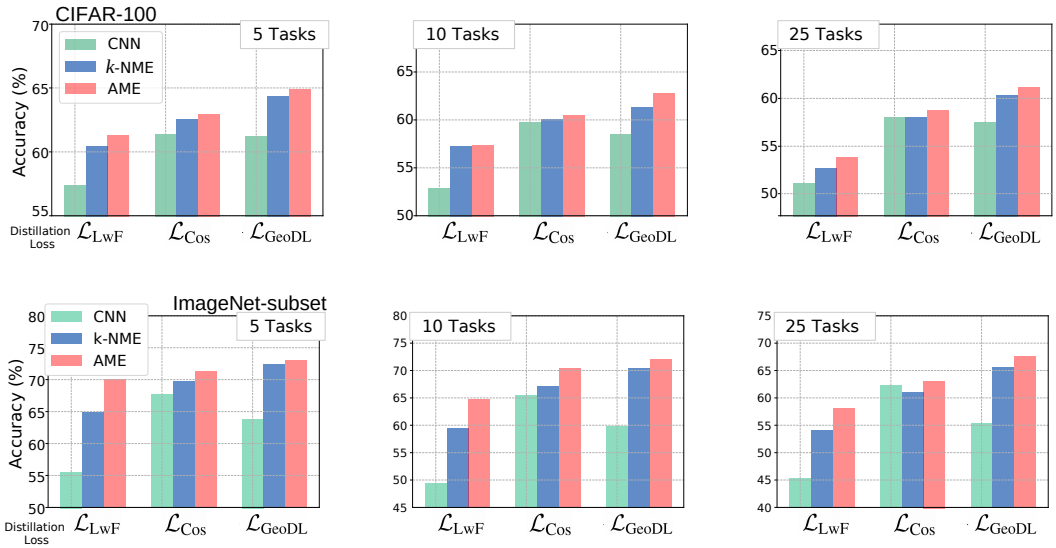


Figure 6.5: The average accuracy on CIFAR-100 (top) and ImageNet-subset (bottom) by varying the number of tasks (5, 10, 25) and classifiers (CNN, k-NME, AME). The objective function contains only cross-entropy and distillation losses. The memory size is 20 exemplars per class.

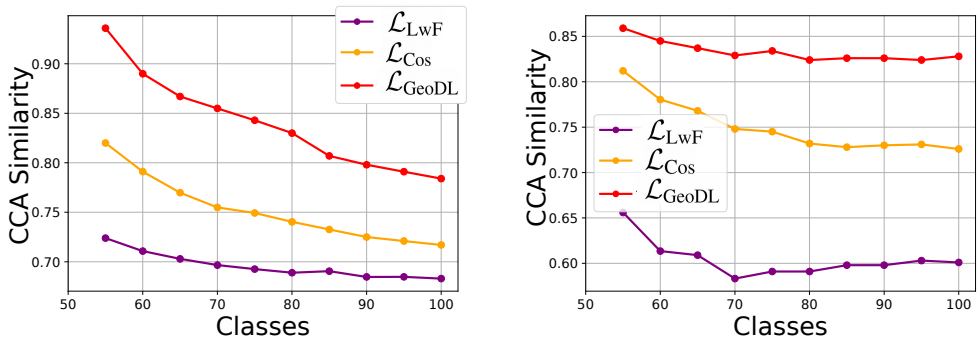
Following work [Hou et al., 2019b], we refer this classifier as CNN. The second classifier is based on selecting the k-nearest neighbors from the samples within the same class (k-NME) [Hou et al., 2019b]. The third classifier is based on the class embedding of all samples (AME) [Hou et al., 2019b]. Fig. 6.5 shows that all distillation losses using AME achieve better accuracy in comparison to other classifiers. Furthermore, our proposed method achieves the best performance given AME. GeoDL improves the performance for IL without any additional losses. We note that our method does not improve the performance of CNN classifier by a large margin because the CNN classifiers trained on past tasks do not get updated when the model learns a new task.

Impact of increasing the number of exemplars in the memory. We investigate the accuracy for 20, 40, 60, 80, 100 exemplars in the memory. In order to investigate the trend of increasing the number of exemplars, only the cross-entropy loss is appended in addition to the baseline ablation setting. Table 6.2 shows that GeoDL outperforms the accuracy of training across various numbers of exemplars in the memory given the LwF [Li and Hoiem, 2017] with the feature distillation loss [Hou et al., 2019b]. The CNN classifier attains higher performance given more exemplars compared to k-NME and AME due to training with a well-balanced number of exemplars per class. We also note that a high number of exemplars (*e.g.*, 100 on CIFAR-100) in the memory helps close the performance gap between training given only the standard cross-entropy loss *vs*. using the additional distillation loss. Table 6.3 shows that our method outperforms the other knowledge distillation techniques under all memory sizes. Using 20 exemplars in the memory, our method $\mathcal{L}_{\text{GeoDL}}$ outperforms the feature distillation loss $\mathcal{L}_{\text{LwF}}$ and the prediction distillation loss $\mathcal{L}_{\text{Cos}}$ by 1.6% and 7.4%, respectively. Unlike on CIFAR-100, AME obtains the highest accuracy under most memory sizes on ImageNet-subset. We also note

Method	Classifier	Average accuracy (%) Memory size per class				
		20	40	60	80	100
$\mathcal{L}_{\text{CE}}$	CNN	50.67	56.63	61.71	62.61	65.91
	k-NME	51.42	56.12	61.11	61.88	64.81
	AME	53.01	56.77	61.64	62.26	64.35
$\mathcal{L}_{\text{CE}} + \mathcal{L}_{\text{LwF}}$ [Li and Hoiem, 2017]	CNN	52.84	58.84	63.14	64.94	67.24
	k-NME	57.31	59.07	63.14	64.35	66.53
	AME	57.41	59.21	63.47	64.57	66.73
$\mathcal{L}_{\text{CE}} + \mathcal{L}_{\text{Cos}}$ [Hou et al., 2019b]	CNN	59.82	61.55	65.39	65.11	67.68
	k-NME	60.09	61.32	64.86	64.34	66.84
	AME	60.46	61.21	64.85	64.32	66.78
$\mathcal{L}_{\text{CE}} + \mathcal{L}_{\text{GeoDL}}$	CNN	58.54	58.72	**66.00**	**66.55**	**68.13**
	k-NME	61.40	61.86	65.82	65.54	67.31
	AME	**62.77**	**62.78**	65.92	65.58	67.38

Table 6.2: The average accuracy for 10 tasks on CIFAR-100 by varying the number of exemplars in the memory.

that using more exemplars in the memory helps close the performance gap between training the model without and with a distillation loss.



(a) CIFAR-100 (b) ImageNet-subset

Figure 6.6: (a) The CCA similarity score between the feature extractor at a specific time $\boldsymbol{\theta}_t$ and the base model $\boldsymbol{\theta}_0$ on CIFAR-100. (b) The CCA similarity score between the feature extractor at a specific time $\boldsymbol{\theta}_t$ and the base model $\boldsymbol{\theta}_0$ on ImageNet-subset. The score is computed based on the feature outputs in the last layer of $\boldsymbol{\theta}_t$ and $\boldsymbol{\theta}_0$.

Impact of subspace dimension on the result. In our method (see PCA in Algorithm 5), the subspace dimension n is a hyperparameter to be tuned. Table 6.4 shows the average accuracy for 10 tasks on CIFAR-100. Our method is robust to the choice of n and we set $n = 127$ in general.

Analyzing the similarity between models. As we observe that different classifiers may yield

Method	Classifier	Average accuracy (%) Memory size per class				
		20	40	60	80	100
$\mathcal{L}_{\text{CE}}$	CNN	46.89	56.53	60.84	63.57	66.06
	k-NME	55.22	61.16	64.26	66.03	68.19
	AME	60.65	64.40	66.72	67.81	69.58
$\mathcal{L}_{\text{CE}} + \mathcal{L}_{\text{LwF}}$ [Li and Hoiem, 2017]	CNN	49.39	57.17	61.74	64.66	66.73
	k-NME	59.37	64.96	67.55	68.71	70.40
	AME	64.67	67.85	69.35	70.00	71.32
$\mathcal{L}_{\text{CE}} + \mathcal{L}_{\text{Cos}}$ [Hou et al., 2019b]	CNN	65.41	68.38	70.52	71.51	72.77
	k-NME	67.05	68.80	70.77	71.14	72.61
	AME	70.38	70.29	71.78	71.65	73.21
$\mathcal{L}_{\text{CE}} + \mathcal{L}_{\text{GeoDL}}$	CNN	59.81	66.09	68.68	70.58	72.04
	k-NME	70.37	72.24	73.32	73.71	**74.28**
	AME	**72.01**	**73.00**	**73.57**	**73.76**	74.25

Table 6.3: The average accuracy for 10 tasks on ImageNet-subset by varying the number of exemplars in the memory.

Subspace Dimension n	**16**	**32**	**64**	**127**
GeoDL	61.87	61.04	61.66	62.77

Table 6.4: Results w.r.t. the subspace dimension (CIFAR-100).

different accuracy for IL tasks. In this experiment, we explore the similarity notion between the current model and the old model using Canonical Correlation Analysis (CCA) in [Raghu et al., 2017], as a tool to analyze the representation of deep models. We evaluate the score based on the current model at a specific time $\boldsymbol{\theta}_t$ and the base model $\boldsymbol{\theta}_0$ with the samples coming from the base classes. The high CCA scores show that the model is *less-forgetting*. Fig. 6.6 (a) shows that the CCA similarity on CIFAR-100 using our approach is the highest compared to training the model with the other distillation losses $\mathcal{L}_{\text{LwF}}$ and $\mathcal{L}_{\text{Cos}}$. We also note that our approach results in the highest CCA similarity between the current feature extractor $\boldsymbol{\theta}_t$ and the base model $\boldsymbol{\theta}_0$, as shown in Fig. 6.6 (b). The high CCA similarity scores indicate that the current model at time t still highly preserves the representations from the base model (evaluated on the samples of the base classes).

6.5 Chapter Conclusion

We have presented a novel distillation loss for IL called GeoDL. In contrast to the prior methods, GeoDL considers the gradual change between consecutive tasks of IL to prevent *catastrophic forgetting*. To this end, our objective function uses the geodesic path between the representations of current task and old task, which results in a smooth transition of the learning process. Our approach achieves competitive results compared to the state of the art for IL on various datasets. Furthermore, GeoDL consistently improves existing baselines and outperforms prior knowledge distillation techniques. The ablation studies also highlight that GeoDL performs better than previous distillation losses for IL.

Chapter 7

Towards a Robust Differentiable Architecture Search under Label Noise

In this chapter, we investigate two research questions: (1) "Is the neural architecture search algorithm affected by the presence of label noise?", (2) "Can we design a robust architecture to learn under label noise?". Our observation shows that the found architecture is far from optimal when searching under label noise. The performance of the optimal architecture also degrades because of noisy labels. We mitigate the aforementioned problems in a single solution by proposing a noise injection module in the neural operation. Furthermore, the proposed module has a profoundly underlying theory in mutual information. The empirical results show that the searched architecture is robust against label noise.

7.1 Introduction

In supervised learning, designing neural network architectures is a noteworthy step towards achieving a competitive performance. To avoid exhausting engineering, Neural Architecture Search (NAS) has emerged as a leading mechanism for automatic design and wiring of neural networks. NAS has been successfully moving forward with diverse approaches to achieve a robust automatic architecture search *e.g.*, evolution-based NAS [Elsken et al., 2019; Real et al., 2017; Stanley et al., 2019; Real et al., 2019], optimization-based NAS [Liu et al., 2019b; Dong and Yang, 2019; Luo et al., 2018; Saxena and Verbeek, 2016; Veniat and Denoyer, 2018], and Reinforcement Learning (RL) based NAS [Zoph and Le, 2017; Zoph et al., 2018; Pham et al., 2018]. Specifically, a gradient-based method with a continuous architecture space called Differentiable ARchiTecture Search (DARTS) has attracted significant attention in NAS because of a reduced cost and complexity of searching for high performance architectures. In this paper, we go beyond merely learning with NAS under regular assumptions of clean labels.

Supervised learning with neural networks often leads to a performance degradation due to overfitting, especially in the presence of label noise which often emerges due to data corruption and/or human annotation errors especially prominent in large scale datasets. As a result, neural networks fail to generalize well to previously unseen data and achieve suboptimal classification results. Given the importance of such problems, existing state-of-the-art methods are specifically

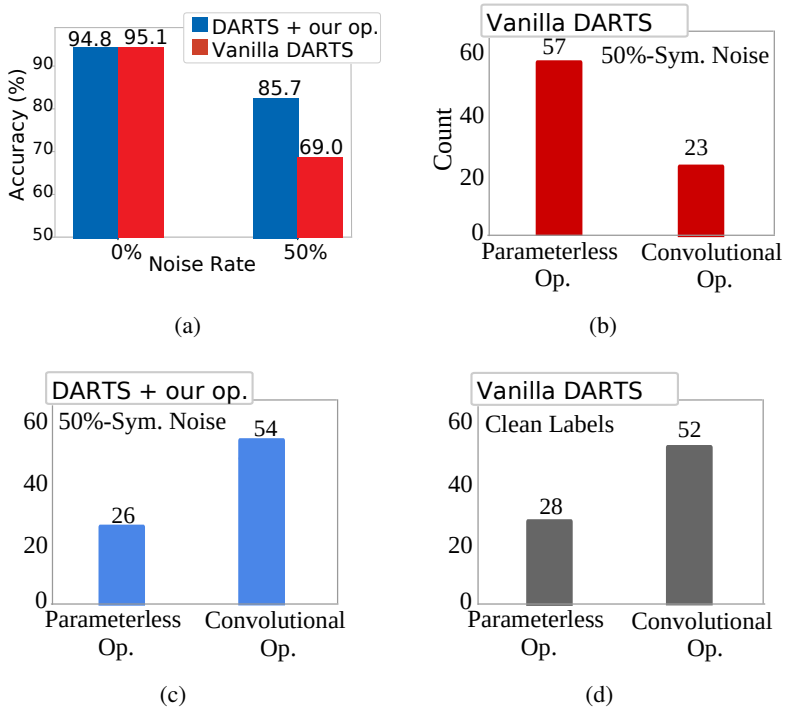


Figure 7.1: (a) Testing accuracy of vanilla *vs.* our approach (CIFAR-10, clean labels *vs.* 50%-symmetric label noise). The histogram of found operators (five runs) of (b) the normal cell of vanilla DARTS under 50%-symmetric label noise, (c) the normal cell of DARTS with nConv (our noise injecting operator) searched on CIFAR-10 (50%-symmetric noisy labels), and (d) the normal cell of vanilla DARTS with clean labels. The normal cell of vanilla DARTS under label noise is constructed poorly due to the large number of parameterless operations. It is apparent the network thus looses the learning capacity in an attempt to prevent overfitting to the noise by selecting parameterless operators. In contrast, DARTS with our proposed operator mitigates such a poor cell design as highlighted by the larger number of convolutional operators being selected in place of a parameterless operator.

designed to deal with the data noise by correcting labels [Vahdat, 2017], employing dedicated loss functions [Arazo et al., 2019; Wang et al., 2019; Natarajan et al., 2013; Zhang and Sabuncu, 2018], reweighting samples [Ren et al., 2018b; Jiang et al., 2018], selecting samples [Han et al., 2018], and modeling a transition matrix [Xia et al., 2019; Patrini et al., 2017; Tanno et al., 2019]. However, designing robust neural networks that mitigate overfitting due to label noise is still unexplored. To this end, we propose a structural approach, that is a method that requires no explicit changes to neither the loss functions nor the input samples nor the final outputs (predictions) but the neural network structure. Moreover, our approach does not require specific assumptions on the amount or the type of label noise.

Our primary focus[1] is to investigate and advance a robust method to search an architecture in the presence of label noise. To this end, we aim to answer the following questions:

The answer to the first question is affirmative as shown in Fig. 7.1. Firstly, we highlight that

[1]Note that training robust NAS under the label noise is not the same problem as using a static network for classification under the label noise.

the performance of vanilla DARTS [Liu et al., 2019b] suffers when subjected to noisy labels as shown in Fig. 7.1 (a). Furthermore, the architecture searched by vanilla DARTS under label noise results in a bad architecture as shown in Fig. 7.1 (b) while our approach shown in Fig. 7.1 (c) produces a better designed cell. To answer the second question, we analyze the task of learning deep neural networks under noisy labels using Information Bottleneck. As a result of this analysis, we introduce a variant of the convolutional operation by injecting noise into the pipeline. Upon learning the parameters of the noise from data, we will empirically show that robustness against label corruption can be attained. In essence, we will show later that the noisy convolution regularizes and limits the gradient of the noisy samples during training. In summary, our key contributions are stated below:

i. We show that the search through noisy labels degrades the classification performance of standard NAS.

ii. We provide an information theoretic framework to tackle learning noisy labels. Our proposed noise injection operator performs implicit regularization during learning preventing overfitting to noisy labels.

iii. The proposed operator is included into the NAS search space with the goal of preventing overfitting during training with noisy labels. A noise injection on the input of the operator turns activations of hidden units into the so-called stochastic activations.

iv. We show experimentally that architectures emerging from the NAS search with our proposed noise injecting operator outperform under fewer parameters the state of the art, especially in the case of no prior knowledge given *e.g.*, the lack of the noise type/its rates.

7.2 Robust Differentiable Architecture Search

A NAS algorithm operates in two stages, namely the **search phase** and the **evaluation phase**. In the search phase, NAS searches the space of cell architectures given a set of operations. During the evaluation phase, a neural network is constructed by stacking multiple cells prior to re-training it from scratch to evaluate the obtained architecture. In the noisy setting, NAS faces incorrect (noisy) labels during both the search and the evaluation phases.

To be more specific, let $\mathcal{D}$ be a dataset with M samples $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{M}$ where $\boldsymbol{x}_i \in \mathbb{R}^n$ and $\boldsymbol{y}_i \in \{0,1\}^C$. We define the noisy data as $\tilde{\mathcal{D}} = \{(\boldsymbol{x}_i, \tilde{\boldsymbol{y}}_i)\}$ where $\tilde{\boldsymbol{y}}_i \in \{0,1\}^C$ is a noisy label. There are two types of label noise that we consider in this chapter, namely symmetric and asymmetric noise. Symmetric noise is constructed by swapping clean labels so that labels of each class are contaminated in a chosen equal proportion by other classes. Asymmetric noise is generated by replacing a chosen percentage of labels of one class with labels of another class *e.g.*, cat $\rightarrow$ tiger. The goal of NAS for noisy labels is to find a robust high-performance architecture that copes well with the label noise during training and testing (prevents overfitting to noisy labels).

7.2.1 Vanilla DARTS

In what follows, we build on the DARTS algorithm [Liu et al., 2019b] which is the driving force behind several recent developments [Lian et al., 2020; Liu et al., 2019a; Xu et al., 2020] due to its significant reduction in computational load of constructing high-performance architectures compared to non-differentiable NAS. Prior to introducing our key contributions, we briefly outline DARTS below.

To perform the search over architectures, DARTS formulates the search phase as a bilevel optimization problem. To this end, an inner and an outer objective function $\mathcal{L}_{\text{trn}}$ and $\mathcal{L}_{\text{val}}$ are introduced with the goal of finding the architecture parameterization $\boldsymbol{\alpha} \in \mathcal{A}$ that minimizes $\mathcal{L}_{\text{val}}$ and the network weights $\boldsymbol{\theta}$ that minimize $\mathcal{L}_{\text{trn}}$ as follows:

$$\begin{aligned} &\min_{\boldsymbol{\alpha}} \ \mathcal{L}_{\text{val}}(\boldsymbol{\theta}^*(\boldsymbol{\alpha}), \boldsymbol{\alpha}) \\ &\text{s.t. } \boldsymbol{\theta}^*(\boldsymbol{\alpha}) = \arg\min_{\boldsymbol{\theta}} \mathcal{L}_{\text{trn}}(\boldsymbol{\theta}, \boldsymbol{\alpha}), \end{aligned} \tag{7.1}$$

where $\boldsymbol{\theta}^*(\boldsymbol{\alpha})$ represents the weights of the associated architecture that minimize the inner loss. Optimizing both losses is achieved by gradient descent. Unfortunately, optimization in the inner loop is time consuming. As such, DARTS applies a simple approximation with a single unrolling step to optimize the inner loss $\nabla_{\boldsymbol{\alpha}} \mathcal{L}_{\text{val}}(\boldsymbol{\theta}^*(\boldsymbol{\alpha}), \boldsymbol{\alpha}) \approx \nabla_{\alpha} \mathcal{L}_{\text{val}}(\boldsymbol{\theta}', \boldsymbol{\alpha})$ where $\boldsymbol{\theta}' = \boldsymbol{\theta} - \eta \nabla_{\theta} \mathcal{L}_{\text{trn}}(\boldsymbol{\theta}, \boldsymbol{\alpha})$. In the bi-level formulation, we encounter two derivatives (related to the network parameters and the architecture parameters, respectively) for which the chain rule is applied together with the implicit function theorem [Larsen et al., 1996; Bengio, 2000; sheng Foo et al., 2008] which yields:

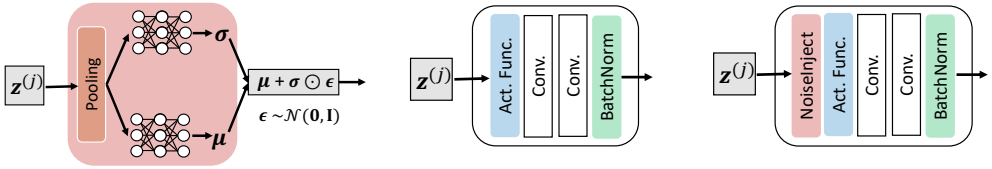
$$\begin{aligned} \nabla_{\boldsymbol{\alpha}} \mathcal{L}_{\text{val}}(\boldsymbol{\theta}^*(\boldsymbol{\alpha}), \boldsymbol{\alpha}) \approx \ & \nabla_{\boldsymbol{\alpha}} \mathcal{L}_{\text{val}}(\boldsymbol{\theta}', \boldsymbol{\alpha}) \\ & - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{val}}(\boldsymbol{\theta}', \boldsymbol{\alpha}) \nabla^2_{\boldsymbol{\theta}, \boldsymbol{\alpha}} \mathcal{L}_{\text{trn}}(\boldsymbol{\theta}, \boldsymbol{\alpha}). \end{aligned} \tag{7.2}$$

To reduce the training time and computational complexity of the algorithm, one can disregard the higher-order derivatives in Eq. 7.2. This simplification leads to a speed-accuracy trade-off as observed in several studies (*e.g.*, [Liu et al., 2019b; Zela et al., 2020a])

Following [Zoph et al., 2018; Real et al., 2019; Liu et al., 2019b], we make use of two types of cells to design the network, namely **I.** the normal cell and **II.** the reduction cell. Each cell is represented by a DAG and can realize one operation from a complete set of candidate operations. The final architecture of the network is constructed by stacking the obtained cells together. This enables us to make the composition of the cells both transferable between datasets and independent of the network depth. A cell is a Directed Acyclic Graph (DAG) with an ordered sequence of N nodes and is connected to its preceding two cells. Let $\mathcal{O}$ be the set of candidate operators (*e.g.*, skip connection, 3×3 dilated convolution, etc.) defined for a cell. Consider a node i whose output represented by $z^{(i)}$ is obtained by:

$$z^{(i)} = \sum_{j \rightsquigarrow i} o^{(i,j)}\left(z^{(j)}\right), \tag{7.3}$$

where $\rightsquigarrow$ denotes the edge between nodes i and j with a total weight over possible operations



(a) (b) (c)

Figure 7.2: (a) The noise injection module which is incorporated into a convolutional operator. (b) The basic convolutional operator used in DARTS (c) Our convolutional operator with additional noise injection at the input $\boldsymbol{h}$.

given as:

$$o^{(i,j)}(\boldsymbol{z}^{(j)}) = \sum_{r=1}^{\nu} \frac{\exp\left(\alpha_r^{(i,j)}\right)}{\sum_{s=1}^{\nu} \exp\left(\alpha_s^{(i,j)}\right)} o_r^{(i,j)}(\boldsymbol{z}^{(j)}) \,, \tag{7.4}$$

where $o_r^{(i,j)}$ are individual operations. In essence, choosing the operation that will connect node j to node i is formulated by a mixing weight vector $\boldsymbol{\alpha}^{(i,j)} = (\alpha_1^{(i,j)}, \alpha_2^{(i,j)}, \cdots, \alpha_\nu^{(i,j)})$ using a softmax function. The task of architecture search is then to learn $\boldsymbol{\alpha} = \left\{\alpha^{(i,j)}\right\}$. At the end of the search phase, a discrete architecture is picked by choosing the most likely operation between the nodes. That is, $o^{(i,j)} = o_{r^*}^{(i,j)}$ where $r^* = \arg\max_r \alpha_r^{(i,j)}$.

7.2.2 Learning in the presence of label noise via the information bottleneck principle

Differentiable NAS uses the cross-entropy loss to train (search and evaluation) networks. Below, we formulate a loss function which prevents the trained model from overfitting to noisy labels. To this end, we bring the information theoretic view and explain how it helps with training under label noise. We begin by defining several terms in our analysis: the input $\boldsymbol{x} \in X$, the output $\boldsymbol{y} \in Y$, the representation of the input $\boldsymbol{z} \in Z$, entropy $H(X) = \mathbb{E}_{p(\boldsymbol{x})}[-\log p(\boldsymbol{x})]$, cross-entropy $H(p,q) = \mathbb{E}_{p(\boldsymbol{x})}[-\log q(\boldsymbol{x})]$, conditional entropy $H(Y|X) = H(X,Y) - H(X)$, mutual information $I(X;Y) = H(Y) - H(Y|X)$, and Kullback-Leibler (KL) divergence $\mathrm{KL}(p(\boldsymbol{x})\|q(\boldsymbol{x})) = \mathbb{E}_{p(\boldsymbol{x})}[\log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}]$.

Considering our NAS setting, we apply the information theoretical framework that includes the mutual information between the learnt representations and labels. We revisit the Information Bottleneck [Tishby et al., 2000] principle which provides a computational framework that controls the trade-off between the compression of input $\boldsymbol{x}$ and the prediction of $\boldsymbol{y}$ according to:

$$\max_{\boldsymbol{\phi}} \; I(Z;Y,\boldsymbol{\phi}) - \beta I(X;Z,\boldsymbol{\phi}), \tag{7.5}$$

where $\beta \geq 0$ is a hyper-parameter adjusting the level of relevant information captured by Z while $\boldsymbol{\phi}$ denotes network parameters. For training neural networks under the regular setting

(clean labels), one maximizes $I(Z;Y,\boldsymbol{\phi})$ to obtain a model with a high performance. However, increasing the value of $I(Z;Y,\boldsymbol{\phi})$ alone makes the model prone to overfitting [Achille and Soatto, 2018] *i.e.*, , the network memorizes the dataset and overfits to wrongly labeled datapoints which results in a performance degradation, as shown in our experiments § 7.3. Let us consider the clean and noisy labels $\hat{Y}$ and $\widetilde{Y}$ separately by splitting the first term in Eq. 7.5 which leads to:

$$\begin{aligned} I(Z;Y,\boldsymbol{\phi}) &= I(Z;\hat{Y},\boldsymbol{\phi}) + I(Z;\widetilde{Y},\boldsymbol{\phi}) \\ &= \int p(\hat{\boldsymbol{y}},\boldsymbol{z}|\boldsymbol{\phi}) \log \frac{p(\hat{\boldsymbol{y}},\boldsymbol{z}|\boldsymbol{\phi})}{p(\hat{\boldsymbol{y}}|\boldsymbol{\phi})p(\boldsymbol{z}|\boldsymbol{\phi})}\, d\hat{\boldsymbol{y}}\, d\boldsymbol{z} \\ &+ \int p(\widetilde{\boldsymbol{y}},\boldsymbol{z}|\boldsymbol{\phi}) \log \frac{p(\widetilde{\boldsymbol{y}},\boldsymbol{z}|\boldsymbol{\phi})}{p(\widetilde{\boldsymbol{y}}|\boldsymbol{\phi})p(\boldsymbol{z}|\boldsymbol{\phi})}\, d\widetilde{\boldsymbol{y}}\, d\boldsymbol{z}. \end{aligned} \tag{7.6}$$

Substituting $p(\boldsymbol{y},\boldsymbol{z}) = p(\boldsymbol{y}|\boldsymbol{z})p(\boldsymbol{z})$ in Eq. 7.6, we obtain:

$$\begin{aligned} I(Z;\hat{Y},\boldsymbol{\phi}) &= \int p(\hat{\boldsymbol{y}},\boldsymbol{z}|\boldsymbol{\phi}) \log \frac{p(\hat{\boldsymbol{y}},\boldsymbol{z}|\boldsymbol{\phi})}{p(\hat{\boldsymbol{y}}|\boldsymbol{\phi})}\, d\hat{\boldsymbol{y}}\, d\boldsymbol{z} \\ I(Z;\widetilde{Y},\boldsymbol{\phi}) &= \int p(\widetilde{\boldsymbol{y}},\boldsymbol{z}|\boldsymbol{\phi}) \log \frac{p(\widetilde{\boldsymbol{y}}|\boldsymbol{z};\boldsymbol{\phi})}{p(\widetilde{\boldsymbol{y}}|\boldsymbol{\phi})}\, d\widetilde{\boldsymbol{y}}\, d\boldsymbol{z}. \end{aligned} \tag{7.7}$$

As estimating terms $p(\widetilde{\boldsymbol{y}}|\boldsymbol{z};\boldsymbol{\phi})$ and $p(\hat{\boldsymbol{y}}|\boldsymbol{z};\boldsymbol{\phi})$ of the mutual information is often computationally intractable, it can be completed with the variational approximation. We note that:

$$\begin{aligned} I(Z;\hat{Y},\boldsymbol{\phi}) &\geq \int p(\hat{\boldsymbol{y}},\boldsymbol{z}|\boldsymbol{\phi}) \log \frac{\bar{p}(\hat{\boldsymbol{y}}|\boldsymbol{z};\boldsymbol{\phi})}{p(\hat{\boldsymbol{y}}|\boldsymbol{\phi})}\, d\hat{\boldsymbol{y}}\, d\boldsymbol{z}, \\ I(Z;\widetilde{Y},\boldsymbol{\phi}) &\geq \int p(\widetilde{\boldsymbol{y}},\boldsymbol{z}|\boldsymbol{\phi}) \log \frac{\bar{p}(\widetilde{\boldsymbol{y}}|\boldsymbol{z};\boldsymbol{\phi})}{p(\widetilde{\boldsymbol{y}}|\boldsymbol{\phi})}\, d\widetilde{\boldsymbol{y}}\, d\boldsymbol{z}. \end{aligned}$$

We formulate the lower bound for the information bottleneck with noisy labels as follows:

$$\begin{aligned} &I(Z;\hat{Y},\boldsymbol{\phi}) + I(Z;\widetilde{Y},\boldsymbol{\phi}) - \beta I(X;Z,\boldsymbol{\phi}) \geq \\ &\int p(\boldsymbol{x})p(\hat{\boldsymbol{y}}|\boldsymbol{x};\boldsymbol{\phi})p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi}) \log \bar{p}(\hat{\boldsymbol{y}}|\boldsymbol{z};\boldsymbol{\phi})\, d\boldsymbol{x}\, d\hat{\boldsymbol{y}}\, d\boldsymbol{z} \\ &+ \int p(\boldsymbol{x})p(\widetilde{\boldsymbol{y}}|\boldsymbol{x};\boldsymbol{\phi})p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi}) \log \bar{p}(\widetilde{\boldsymbol{y}}|\boldsymbol{z};\boldsymbol{\phi})\, d\boldsymbol{x}\, d\widetilde{\boldsymbol{y}}\, d\boldsymbol{z} \\ &-\beta \int p(\boldsymbol{x})p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi}) \log \frac{p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})}{p(\boldsymbol{z}|\boldsymbol{\phi})}\, d\boldsymbol{x}\, d\boldsymbol{z} = \mathcal{L}. \end{aligned} \tag{7.8}$$

Let $q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})$ and $r(\boldsymbol{z})$ be a variational approximation for $p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})$ and $p(\boldsymbol{z}|\boldsymbol{\phi})$, then we have the lower bound:

$$\begin{aligned} \mathcal{L} \approx \frac{1}{|\hat{Y}|+|\widetilde{Y}|} \sum_{n=1}^{|\hat{Y}|+|\widetilde{Y}|} &\mathbb{E}_{z\sim q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})}\left[\log p(\boldsymbol{y}_n|\boldsymbol{z})\right] \\ &-\beta \mathrm{KL}\left[q(\boldsymbol{z}|\boldsymbol{x}_n;\boldsymbol{\phi})||r(\boldsymbol{z})\right], \end{aligned} \tag{7.9}$$

where $q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})$ denotes our variational approximation using the reparameterization trick [Kingma

and Welling, 2013; Alemi et al., 2017]. Given the case with noisy labels, we can reduce $I(Z;\widetilde{Y},\boldsymbol{\phi})$ by controlling the first term in Eq. 7.9 to avoid overfitting in the training phase. In fact, we can control the first term by adjusting the noise variance of the second term of Eq. 7.9 as follows.

Suppose an encoder is used for $q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})$, $r(\boldsymbol{z}) = \mathcal{N}(\mathbf{0},\mathbf{I})$ and $\boldsymbol{\phi} = \{\boldsymbol{\phi}_1,\boldsymbol{\phi}_2\}$. We can calculate the KL divergence in the closed form as follows:

$$\mathrm{KL}(q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})||r(\boldsymbol{z})) = -\sum_{j}^{J}\log\sigma_j{}^2 + \frac{\sigma_j{}^2+\mu_j{}^2}{2} - \frac{1}{2}, \tag{7.10}$$

where $\boldsymbol{z}\sim\mathcal{N}(\boldsymbol{\mu}_x,\boldsymbol{\sigma}_x^2) = q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})$, $\boldsymbol{\mu}_x = f_{\boldsymbol{\phi}_1}(\boldsymbol{x})$, $\boldsymbol{\sigma}_x = f_{\boldsymbol{\phi}_2}(\boldsymbol{x})$, and j is the index element of $\boldsymbol{\mu}_x$ and $\boldsymbol{\sigma}_x$.

7.2.3 DARTS meets the label noise

The variational information bottleneck provides an inspiration for our noise injecting operators for differentiable NAS. It helps us limit overfitting and memorization to noisy labels. We argue that the designed operator should exhibit the variational approximation in Eq. 7.9 to adjust $I(Z;\widetilde{Y})$. To this end, a robust operator should inject the right levels of noise to successfully train NAS in the presence of label noise. Thus, we propose a new operator, the noise injecting Convolution, *nConv* for short. In NAS, the architecture parameters $\boldsymbol{\alpha}$ play an important role in assigning weights of operators. As we train the model under label noise, the larger the weights corresponding to our noise injecting operator are, the stronger the regularization effect becomes.

We define the functionality of our nConv operator on its input $\boldsymbol{z}$ as $\boldsymbol{z}\sim\mathcal{N}(\boldsymbol{\mu}_x,\boldsymbol{\sigma}_x^2)$[2]. By properly scaling the noise parameters, we expect a neural network with nConv to exhibit robustness against noisy labels. To this end, we propose to learn the noise parameters (*i.e.*, , $\boldsymbol{\mu}_x$ and $\boldsymbol{\sigma}_x^2$) based on the data distribution. Drawing inspiration from the reparameterization trick [Kingma and Welling, 2013], this is achieved by learning $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ through a network parameterized by $\boldsymbol{\phi}$ as $\boldsymbol{z}\sim q_(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})$ (see Fig. 7.2 (a) for a conceptual diagram). In DARTS, $\boldsymbol{\phi}$ corresponds to noise parameters of all instantiated noise injecting operators. Let us assume $\boldsymbol{\phi}$ is contained within $\boldsymbol{\theta}$ represents the entire learnable set of parameters of all instantiated operators. As derived in Eq. 7.9, the problem of learning the parameters of noise is cast as marginalizing the negative log-likelihood over the data with a KL divergence regularization term as:

$$\begin{aligned}\mathcal{L}_{\mathrm{NAS}} = \sum_{(\boldsymbol{x},\tilde{\boldsymbol{y}})\in\tilde{D}} & -\mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\alpha};\boldsymbol{\theta})}\left[\log p\left(\tilde{\boldsymbol{y}}|\boldsymbol{x};\boldsymbol{\theta};\boldsymbol{\alpha};\boldsymbol{z}\right)\right] \\ & + \beta\mathrm{KL}\left[q\left(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\alpha};\boldsymbol{\theta}\right)||\mathcal{N}\left(\mathbf{0},\mathbf{I}\right)\right].\end{aligned} \tag{7.11}$$

Note that, minimizing Eq. 7.11 also applies when we retrain the network in the evaluation phase by using a discrete version of $\boldsymbol{\alpha}$. Algorithm 7 details the steps of performing DARTS inclusive of our nConv operator.

In vanilla DARTS, two blocks of convolution, namely *Separable Convolution* (SepConv)

[2]We have modeled the Normal distribution by a diagonal covariance. However, our algorithm is generic and can deal with full covariance matrices. We denote the diagonal elements of the covariance matrix by $\boldsymbol{\sigma}^2$.

Algorithm 7 DARTS + nConv

Operation $\mathcal{O}$, arc. parameters $\boldsymbol{\alpha}$, network parameters $\boldsymbol{\theta}$, noise injection parameters $\boldsymbol{\phi}$, dataset $\tilde{\mathcal{D}}$

1: **Phase 1: Search an architecture**
2: **while** not converged **do**
3: Update $\boldsymbol{\alpha}$ using Equation 7.2
4: Update $\boldsymbol{\theta}, \boldsymbol{\phi}$ using $\nabla_{\boldsymbol{\theta},\boldsymbol{\phi}} \mathcal{L}_{\text{NAS}}$
5: **Phase 2: Evaluate an architecture**
6: Construct a final architecture from $\boldsymbol{\alpha}$
7: Reinitialize $\boldsymbol{\theta}, \boldsymbol{\phi}$
8: **while** not converged **do**
9: Update $\boldsymbol{\theta}, \boldsymbol{\phi}$ using $\nabla_{\boldsymbol{\theta},\boldsymbol{\phi}} \mathcal{L}_{\text{NAS}}$

Separable Convolution 3x3 (*sep_conv_3x3*)	Separable Noisy Convolution 3x3 (*sepconv3x3_noise*)
-	Noise injection
ReLU	ReLU
3x3 convolution, C channels, no bias	3x3 convolution, C channels, no bias
1x1 convolution, C channels, no bias	1x1 convolution, C channels, no bias
Batch normalization	Batch normalization
-	Noise injection
ReLU	ReLU
3x3 convolution, C channels, no bias	3x3 convolution, C channels, no bias
1x1 convolution, C channels, no bias	1x1 convolution, C channels, no bias
Batch normalization	Batch normalization

Table 7.1: The Separable Convolution (*SepConv*) and the Separable Noisy Convolution (*SepNConv*).

and *Dilated Convolution* (DilConv) are used to construct the search space (see Table 7.1 and 7.2). Both blocks share the same architectural design (see Fig. 7.2 (b)). The variants of these two blocks are constructed by adjusting the kernel size, the dilation size, and the stride size of the underlying convolutions. We extend this basic block by injecting noise at the input $\boldsymbol{z}^{(j)}$ (see Fig. 7.2 (c)). We can readily extend both blocks with nConv to attain *SepNConv* and *DilNConv* blocks. At the cell level, both normal and reduction cells may include nConv in their structures. In the normal cell, the feature map is generated with the same dimension as the input. For the reduction cell, the input to the noise injection module is applied with a stride of two to shrink the cell output size by a factor of two. Selecting nConv in the cells encourages stochastic activations (the output of hidden units) that help regularize the neural network.

Note that we experimentally observe that multiplication of noise and hidden units (*e.g.*, dropout) in nConv does not work well to tackle the label noise problem. Thus, our noise injection function is defined by addition of noise to the hidden units. The cell output is constructed by combining the output hidden states of all nodes in each cell.

Dilated convolution 3x3 (*dil_conv_3x3*)	Dilated Noisy Convolution 3x3 (*dilconv3x3_noise*)
-	Noise injection
ReLU	ReLU
3x3 convolution, C channels, no bias, dilation 2	3x3 convolution, C channels, no bias, dilation 2
1x1 convolution, C channels, no bias	1x1 convolution, C channels, no bias
Batch normalization	Batch normalization

Table 7.2: The Dilated Convolution (*DilConv*) and the Dilated Noisy Convolution (*DilNConv*).

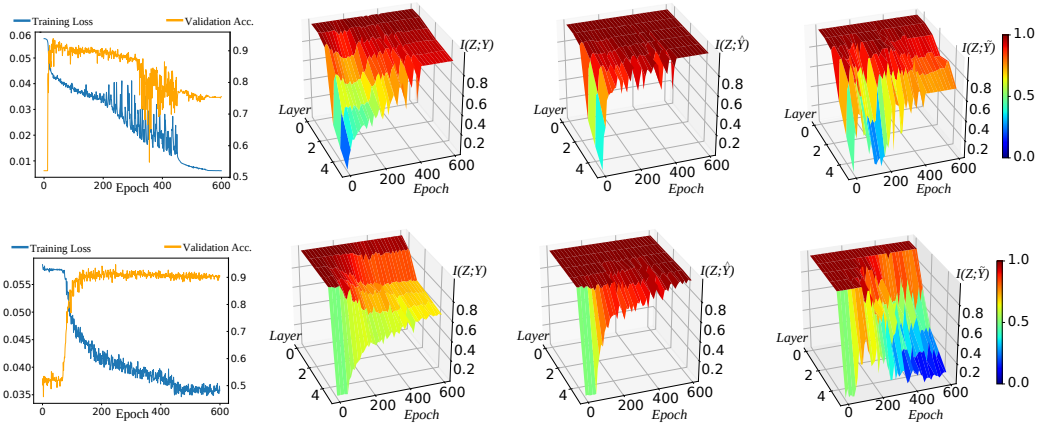


Figure 7.3: Visualization of mutual information for noisy labels. (Top) Training under 20% label noise without (top) and with our approach (bottom). (a) Training loss and validation accuracy across 600 epochs. (b) Mutual information of all labels and the hidden units ($I(Z;Y)$). (c) Mutual information of clean labels with hidden units. (d) Mutual information of noisy labels with hidden units. The higher the value of $I(Z;Y)$, the more information about Y is preserved by variable Z.

7.3 Experiments

7.3.1 The dynamics of training under label noise

In this experiment, we estimate $I(Z;Y)$ to analyze the mutual information between the hidden units of neural networks (with and w/o noise injecting operators) and their respective labels. The settings are adopted from [Shwartz-Ziv and Tishby, 2017] which provides a toy dataset and a technique to visualize the mutual information. The network for this experiment has six layers in total with Tanh activations on intermediate layers and a Sigmoid function for final outputs. We contaminate 20% training data from the toy dataset [Shwartz-Ziv and Tishby, 2017] that has only two categories. Fig. 7.3 shows that training the network with our approach ($\beta = 1$) under label noise is more stable across 600 epochs. The mutual information is indicative of a correlation between two variables. Fig. 7.3 (d) shows that a neural network w/o noise injecting operator overfits to noisy labels (top) while our proposed noise injection reduces overfitting (bottom). This experiment confirms our theoretical analysis that the variational information bottleneck, in essence, reduces the mutual information between the hidden units and noisy labels to combat memorization of corrupted labels.

7.3.2 Searching the architecture

We search an architecture with the following operators $\mathcal{O}$: 3×3 separable convolution, 3×3 dilated convolution, 3×3 separable convolution with noise, 3×3 dilated convolution with noise, 3×3 max pooling, 3×3 average pooling, and the identity. Separable and dilated convolution follows the operations in [Zoph et al., 2018; Real et al., 2019; Liu et al., 2019b]. The architecture search is run on CIFAR-10 consisting of training and validation sets. The setup is similar to DARTS in that there are 7 nodes for each cell and 16 initial filters. We use the momentum SGD, we set the learning rate and the momentum to 0.025 and 0.9, resp., and the weight decay to $3\times\ 10^{-4}$. Moreover, a cosine schedule is employed to anneal the learning rate. A network with 8 cells is trained for 50 epochs with a batch size of 96. The search time is 6 hours on a NVIDIA Titan V GPU.

7.3.3 Datasets and implementations details

The evaluation phase follows the protocol in DARTS in which the architecture is found via a small task and then the cell structure is transferred to other (bigger) tasks. Normal cells and reduction cells are based on the architecture found on CIFAR-10, then the cells are stacked for evaluation. All experiments for the architecture evaluation are averaged over 3 runs. The model is trained for 300 epochs for all datasets. In the evaluation phase, the cells are stacked to form a group of 16 cells and 16 initial filters. The optimizer parameters are set following the setup during the search phase.

Benchmark datasets for robustness evaluation used in our experiments are CIFAR-10 and CIFAR-100 consisting of 50k samples and 10k samples for training and testing, respectively. The image size of both CIFAR datasets are 32×32 following the standard protocol in past NAS and noisy label works [Han et al., 2018; Liu et al., 2019b]. We investigate both symmetric and asymmetric label noise. Symmetric noise rates on CIFAR-10 and CIFAR-100 are 20% and 50%, and the asymmetric noise rate is 40%. All results are reported based on the average of the last ten epochs with $\beta = 1$. We also run evaluations on a larger Tiny-ImageNet[3] with 200 classes and 120k images in total. The image size of Tiny-ImageNet is set to 64×64 following the setup in [Yu et al., 2019]. Finally, we experiment with the Clothing1M [Xiao et al., 2015] which contains real-world noisy labels, 14 categories, and $\sim$1M images (224×224 size). The network is trained for 80 epochs. SGD with momentum is used to optimize the model with the learning rate set to 0.1. Remaining parameters follow the setup for CIFAR.

7.3.4 Comparison with Vanilla DARTS

Below, we compare the performance of Vanilla DARTS and our DARTS with the nConv operator (DARTS + nConv) to study the improvement of the latter when learning in the presence of label noise. We use DARTS with following operators $\mathcal{O}$: 3×3 separable convolution, 3×3 dilated convolution, 5×5 separable convolution, 5×5 dilated convolution, 3×3 max pooling, 3×3 average pooling, and the identity. We use the same setup (*e.g.*, optimizer parameters/batch size) for both techniques to search and evaluate the found architecture. The search/evaluation phases use the same label noise rates (*e.g.*, searching on CIFAR-10 with 50% noise and evaluating

[3] https://tiny-imagenet.herokuapp.com/

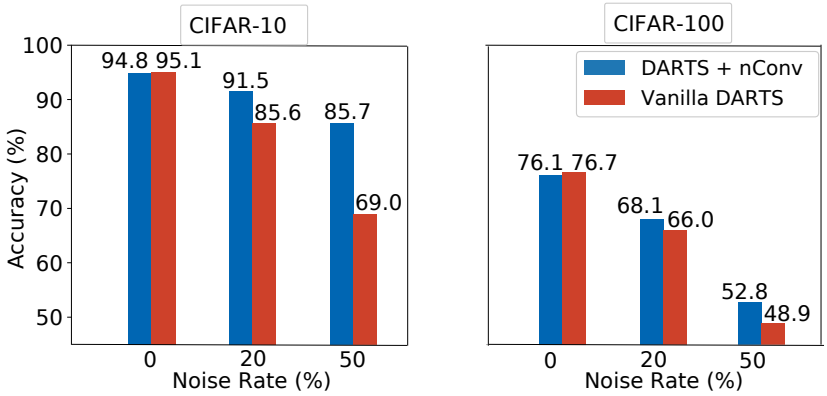


Figure 7.4: Neural architecture search under noisy labels with DARTS and DARTS + nConv on CIFAR-10 and CIFAR-100. The comparison uses three protocols: clean, 20%, and 50% symmetric label noise.

the found architecture on CIFAR-100 with 50% noise). Fig. 7.4 shows that DARTS + nConv achieves superior performance compared to Vanilla DARTS.

Method	**Backbone**	**20%-Sym**	**50%-Sym**
F-Correction [Patrini et al., 2017]	Conv-9 (Size: 4.4M)	84.6	59.8
Decoupling [Malach and Shalev-Shwartz, 2017]		80.4	51.5
MentorNet [Jiang et al., 2018]		80.8	71.1
Co-Teaching [Han et al., 2018]		82.3	74.0
JoCoR [Wei et al., 2020]		85.7	79.41
Cross Entropy	ResNet-18 (Size: 11.2M)	85.6	57.8
F-Correction [Patrini et al., 2017]		83.1	59.4
Decoupling [Malach and Shalev-Shwartz, 2017]		79.9	52.2
MentorNet [Jiang et al., 2018]		80.5	70.7
Co-Teaching [Han et al., 2018]		82.4	72.8
T Revision [Xia et al., 2019]		89.6	83.4
DARTS + nConv	Auto (Size: 1.2M)	**91.4**	**85.7**

Table 7.3: CIFAR-10 test accuracy (%) using ResNet and Conv-9.

7.3.5 Comparison with the state of the art for training under label noise

We compare our method to multiple baselines that are designed to tackle problems resulting from label noise. The baselines we compare to, use three different backbones: Conv-9 [Han et al., 2018], ResNet-18, and ResNet-34 [He et al., 2016]. Table 7.4 shows the results on CIFAR-10 and CIFAR-100 in the presence of 20% as well as 50% symmetric noise. Our proposed method with nConv outperforms T-Revision [Xia et al., 2019] on all protocols by a substantial margin of 2%. In addition, Table 7.5 shows that our proposed method consistently performs better than the state of the art *i.e.*, , joint-optimization [Tanaka et al., 2018], PENCIL [Yi and Wu, 2019], and F-Correction [Patrini et al., 2017]. Note that our approach does not need any modifications of the loss function to train the model under symmetric or asymmetric noise.

We further evaluate the performance on challenging datasets, namely Tiny-ImageNet and

Method	Backbone	20%-Sym	50%-Sym
F-Correction [Patrini et al., 2017]	Conv-9 (Size: 4.4M)	61.9	41.0
Decoupling [Malach and Shalev-Shwartz, 2017]		44.5	25.8
MentorNet [Jiang et al., 2018]		52.1	39.0
Co-Teaching [Han et al., 2018]		54.2	41.4
JoCoR [Wei et al., 2020]		53.0	43.5
Cross Entropy	ResNet-34 (Size: 20.1M)	61.2	41.2
F-Correction [Patrini et al., 2017]		61.4	37.3
Decoupling [Malach and Shalev-Shwartz, 2017]		52.1	38.5
MentorNet [Jiang et al., 2018]		80.5	70.7
Co-Teaching [Han et al., 2018]		54.2	41.4
T Revision [Xia et al., 2019]		65.4	50.5
DARTS + nConv	Auto (Size: 1.2M)	**68.1**	**52.8**

Table 7.4: CIFAR-100 test accuracy (%) using ResNet and Conv-9.

Method	Backbone	Accuracy (%)
Cross-entropy	ResNet-18 (Size: 11.2M)	72.3
F-Correction [Patrini et al., 2017]		83.1
DARTS + nConv	Auto (Size: 1.2M)	**89.5**

Table 7.5: Testing accuracy (%) with 40% asymmetric noise on CIFAR-10.

Clothing1M [Xiao et al., 2015]. Note that our evaluation does not use pre-trained neural networks for both datasets. As shown in Table 7.6, DARTS + nConv also outperforms Co-teaching+ [Yu et al., 2019], Co-teaching [Han et al., 2018], MentorNet [Jiang et al., 2018], and F-Correction [Patrini et al., 2017]. When the noise rate increases from 20% to 50%, the performance gap is widened. As an example, the performance gap increases by 3% when comparing our method with Co-teaching [Han et al., 2018]. Moreover, Table 7.7 shows that our method outperforms state-of-the-art methods on the Clothing1M dataset by 1.3%.

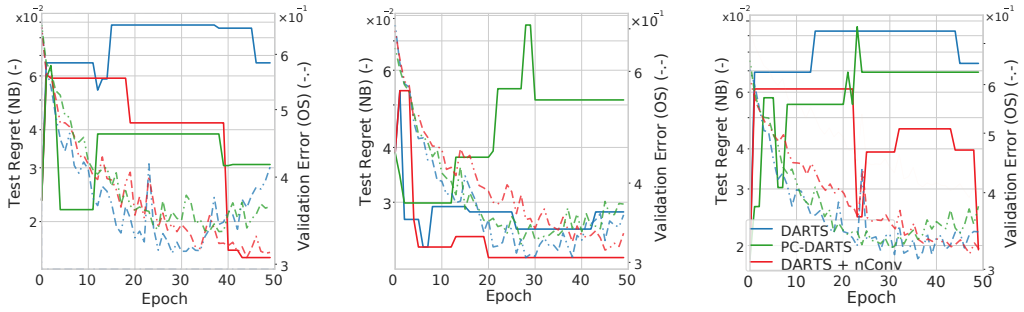
Our proposed method outperforms current state of the art on all datasets irrespective of noise rates while substantially reducing the number of parameters compared to Conv-9, ResNet-18, and ResNet34 backbones. The searched architecture has fewer parameters (size reduced by 10-20$\times$) compared to ResNet backbones. Finally, our approach maintains only one neural network while previous methods (*e.g.*, Decoupling [Malach and Shalev-Shwartz, 2017] and Co-teaching [Han et al., 2018]) train two networks simultaneously.

7.3.6 NAS benchmark evaluation

Below, we examine the capability of DARTS to obtain high performance architectures in the presence of label noise. We compare our method with Vanilla DARTS and PC-DARTS [Xu et al., 2020] using NAS-benchmark 1Shot1 [Zela et al., 2020b]. There are three different search spaces (1,2, and 3) used in our evaluation which differ by the number of nodes and node parents. The search space 3 in this benchmark is the largest search space ($\sim$360,000 architectures) among NAS benchmarks. The detailed setup is provided in our supplementary material. CIFAR-10 is used to search the architecture with 50% symmetric noise. Fig. 7.5 shows that our method does not overfit to the noisy labels and performs better (lower test regret) compared to Vanilla

Method	Backbone	20%-Sym	50%-Sym
Decoupling [Malach and Shalev-Shwartz, 2017]	ResNet-18 (Size: 11.2M)	36.3	22.6
F-Correction [Patrini et al., 2017]		44.4	32.8
MentorNet [Jiang et al., 2018]		45.5	35.5
Co-teaching [Han et al., 2018]		45.6	37.1
Co-teaching+ [Yu et al., 2019]		47.7	41.2
DARTS + nConv	Auto (Size: 1.2M)	**50.6**	**45.7**

Table 7.6: Testing accuracy (%) with 20% and 50% symmetric noise on Tiny-ImageNet.



(a) (b) (c)

Figure 7.5: Test regret of NAS benchmark 1shot1 on CIFAR-10 (searching with 50% symmetric noise) using three search spaces: (a) search space 1, (b) search space 2, and (c) search space 3.

DARTS and PC-DARTS.

7.4 Searching architectures

In Fig. 7.8, we show a cell architecture found via Vanilla DARTS with 20% and 50% symmetric noise on CIFAR-10. Most of the operations in the reduction cells for both noise cases have no parameters (*e.g.*, skip connection and max-pooling). The parameterless cells in Fig. 7.8 have a smaller parameter size compared to the cells found with our operator in Fig. 7.9. For instance, an architecture searched with Vanilla DARTS (16 stacked cells) has 0.6M parameters but an architecture including nConv (16 stacked cells) has 1.2M parameters. It appears that in the presence of the label noise, the standard architecture search engine tries to prevent overfitting by selecting parameterless operations which are not expressive enough and thus lead to sub-optimal architectures with poor performance.

7.5 Ablation Study

7.5.1 The impact of the standard deviation

Below, we perform an ablation study to investigate the impact of different values for standard deviation of the noise injection parameter. The setup for this experiment follows the setup in § 7.3 with the toy data from [Shwartz-Ziv and Tishby, 2017] contaminated with 20% symmetric noise and the six layers neural network. We vary the standard deviation in range $\{0.01, 0.05, 0.2, 0.3\}$ and the mean in range $\{0.1, 0.2, 0.3\}$, and train the model for 600 epochs.

Dataset	DARTS + nConv	Co-Teaching [Han et al., 2018]	Decoupling [Malach and Shalev-Shwartz, 2017]	F-Correction [Patrini et al., 2017]
Clothing1M	**69.8**	68.5	67.3	65.4

Table 7.7: Test accuracy on the Clothing1M dataset. The prior methods use ResNet-18.

Fig. 7.6 shows that the standard deviation affects both the accuracy and the training loss when training the model. We observe that training without the noise injection performs poorly because the model overfits to the noisy labels given the lowest training loss attained. We found that injecting noise with standard deviation of 0.2 or 0.3 yields the highest validation accuracy while avoiding overfitting (indicated by the higher and smoothly decreasing training loss). In this experiment, the improvement using the noise injection module yields 18% improvement over the baseline which does not include the noise injection unit.

We also observe that varying the mean does not affect the performance significantly, thus, we conveniently set the mean to zero for all experiments in this chapter.

7.5.2 Gradient analysis under training with noisy labels

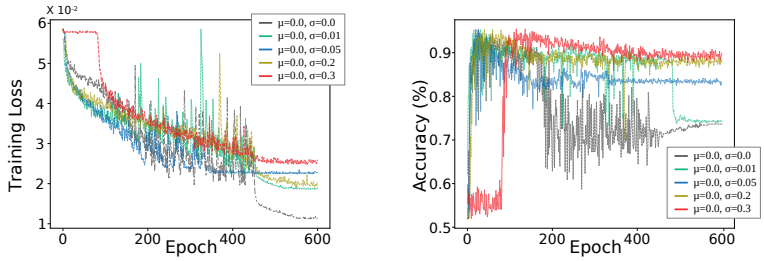
In this experiment, we analyze the impact that training samples with clean labels *vs.* noisy labels has on the gradient. This helps us understand from which samples the network learns and how overfitting occurs due to the label noise.

Fig. 7.7 (a) shows the gradient norm as a function of epoch for Vanilla DARTS. The figure shows that in the early epochs, samples with clean labels dominate over samples with noisy labels, whereas in the later epochs of the training, the reverse is true. This phenomenon leads to a performance drop towards the end of training and the model parameters are comparatively more influenced by the noisy samples rather than the clean samples as the network tries to fit to datapoints with erroneous labels.

We then perform the equivalent experiment using our proposed method, shown in Fig. 7.7 (b). The figure shows that the impact of the gradient of the clean samples dominates the impact of the gradient of the noisy samples throughout the entire training. This illustrates that the risk of overfitting to noisy labels is substantially reduced, and that the network preferentially learns from the clean samples. This observation of how gradient behave translates into the superior performance (under noisy labels) of DARTS + nConv compared to Vanilla DARTS.

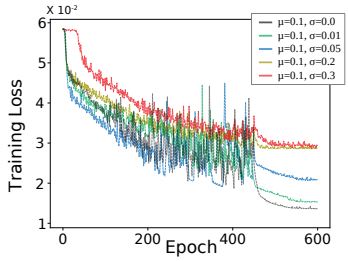
7.6 Chapter Conclusion

We have presented a new operator coined **nConv** to search and train a robust neural network in the presence of label noise. We have shown that label noise harms the performance of Vanilla DARTS while DARTS + nConv performs robustly on problems with noisy labels. The proposed operator has a minimum amount of overhead compared to the existing operators in the search space (*e.g.*, convolution operations in Vanilla DARTS). As changes are applied to the neural network structure, mechanisms such as sample selection, modified loss functions or transition matrices are not needed by our approach. Our empirical results also show that we can design a neural network with fewer parameters compared to ResNet backbones, without the loss of accuracy.

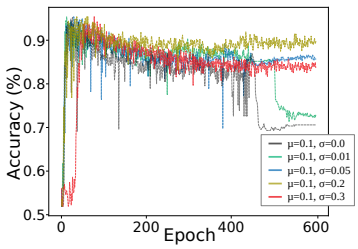


(a) Train ($\mu = 0.0$)

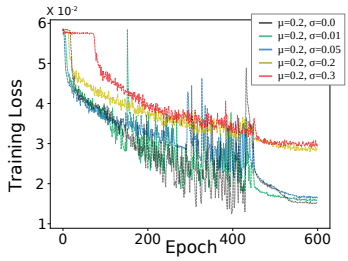
(b) Validation ($\mu = 0.0$)



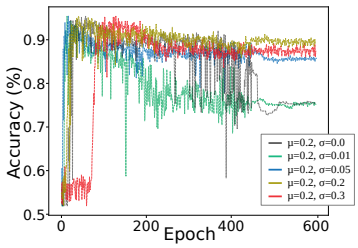
(c) Train ($\mu = 0.1$)



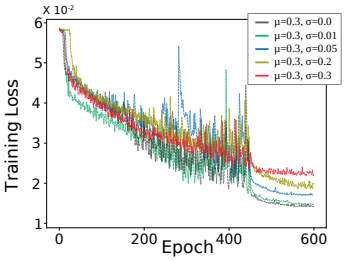
(d) Validation ($\mu = 0.1$)



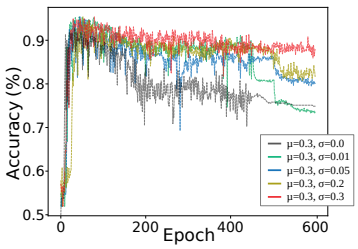
(e) Train ($\mu = 0.2$)



(f) Validation ($\mu = 0.2$)

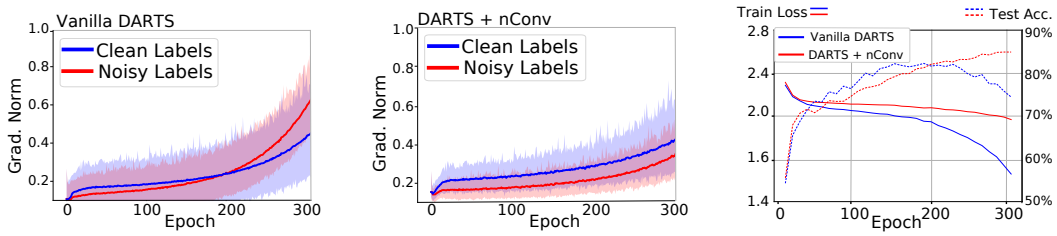


(g) Train ($\mu = 0.3$)



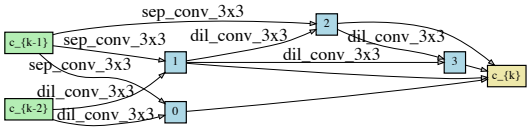
(h) Validation ($\mu = 0.3$)

Figure 7.6: Training under 20% symmetric noise on the toy data by varying the standard deviation according to $\{0.0, 0.01, 0.05, 0.2, 0.3\}$ and the mean according to $\{0.0, 0.1, 0.2, 0.3\}$.

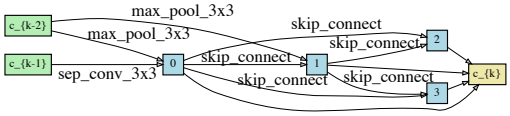


(a) (b) (c)

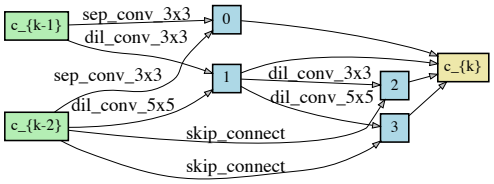
Figure 7.7: The norm of gradients (the mean with standard deviation) while training the found architectures (evaluation phase) on CIFAR-10 with 50%-symmetric label noise. (a) Vanilla DARTS is highly impacted by the gradients of samples with noisy labels after 200 epochs. (b) DARTS + nConv maintains the norm of gradients from both (noisy and clean) samples. (c) Train loss and validation accuracy in the evaluation phase of NAS as a function of epoch number.



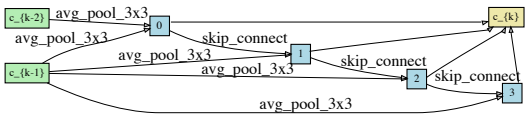
(a) Normal cell (20%-sym)



(b) Reduction cell (20%-sym)

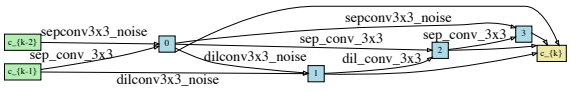


(c) Normal cell (50%-sym)

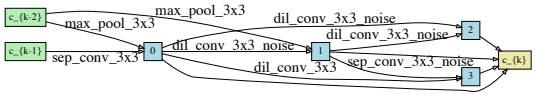


(d) Reduction cell (50%-sym)

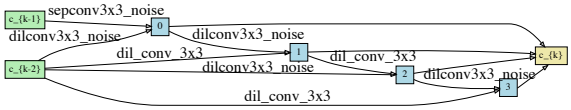
Figure 7.8: Architectures found via Vanilla DARTS with 20% (top) and 50% (bottom) symmetric noise. Figures (a) and (c) show normal cells and figures (b) and (d) show reduction cells. The reduction cells consist of many parameterless operations.



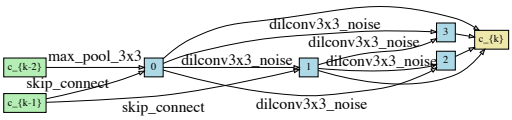
(a) Normal cell (20%-sym)



(b) Reduction cell (20%-sym)



(c) Normal cell (50%-sym)



(d) Reduction cell (50%-sym)

Figure 7.9: Architectures found via DARTS + nConv with 20% (top) and 50% (bottom) symmetric noise. Figures (a) and (c) are normal cells and figures (b) and (d) are reduction cells. The normal and reduction cells consist of noise convolution.

Chapter 8

Conclusions

Generalization and open-ended learning are long-standing problems in machine learning. Along this line of research, this thesis focuses on the problems of few-shot learning, meta-learning, incremental learning, and NAS. Though the development of adaptive deep neural networks has shown some progress, the efficient learning process as in the human brain remains a difficult quest in practice. In this final chapter, we summarize our works, identify its limitations, and discuss the future challenges of adaptation in DNNs.

8.1 Summary

In this thesis, we conducted some work to pursue adaptive capabilities in DNNs. The motivation of adapting DNNs begins with the requirements to build artificial general intelligence that can execute various tasks. In pursuing this ultimate goal, we introduce solutions to a range of problems for neural adaptation. Particularly, our works are important in contributing efficient learning mechanism after deployment. We propose a dynamic classifier for few-shot learning using subspace methods, where a model needs to adapt based on the context in each *episode*. As an extension to improve the performance, the subspace approach is also applied not only for the classifier but also for the query, and the notion of distance on Grassmann manifolds is used to calculate the distance between two subspaces. In a similar problem with few-shot learning, we notice that learning from a few data needs to be quick and avoid overfitting at the same time. To this end, we employ a non-linear function to modulate gradients for fast adaptation of DNNs. Next, we observe that the current problem of few-shot learning is limited to single label instances. This observation leads to the solution for multi-label few-shot learning using multi-label propagation. In the research of incremental learning, a knowledge distillation method is proposed to minimize the dissimilarity between responses of two different tasks where the connection is made by the geodesic path. Finally, to investigate how robust the neural architecture method is for a specific task, we consider the problem of NAS under label noise. In this problem, the empirical results show that the differentiable NAS algorithm is highly affected by noisy labels, as shown by the large number of parameterless operations. We employ the noise injection technique in the operations such that it can mitigate the found architecture to be parameterless. The key contributions of our thesis are summarized below:

- **Chapter 3** proposes adaptive subspaces for few-shot learning including the point-to-subspace approach and the subspace-to-subspace approach using the distance on Grass-

mann manifolds.

- **Chapter 4** proposes a multi-label propagation method and a neural label counting module for multi-label few-shot learning.
- **Chapter 5** proposes a meta-learning strategy for fast adaptation using non-linear function, which is also proven robust against the noise on gradients.
- **Chapter 6** proposes a knowledge distillation method by preserving the projected features from the old model and the current model to align on subspaces along the geodesic flow.
- **Chapter 7** proposes the noise injection module to prevent the differentiable NAS algorithm yielding a poorly constructed architecture when the architecture is searched under label noise.

8.2 Limitations

In few-shot learning, the problem setting is limited to a very low number of classes (*e.g.*, 5-way and 10-way), while in reality, the number of classes can be thousands (*e.g.*, ImageNet [Krizhevsky et al., 2012]). This condition makes the solution less practicable to the challenging problems in computer vision and natural language processing that require a lot of categories. As a comparison, the performance of a few-shot learning method on the 5-way protocol degrades 25% when the same method is evaluated on the 20-way protocol.

In incremental learning, *catastrophic forgetting* is still obvious when the model needs to learn from a long sequence. The long term learning process is troublesome when the storage capacity is limited and the model needs to learn new data sequentially. Furthermore, incremental learning methods mostly aim to solve the problem of adding new classes. However, data in real world scenarios is very complex, where new data can come with various domains, new concepts, input changes, and label shifts. The desiderata of incremental learning also include *forward transfer i.e.*, the previously learned tasks improve the efficiency and the performance of future related tasks. This property is still not investigated in incremental learning, and most of the published works focus on the *catastrophic forgetting* problem.

In NAS, we identify that the architecture has a form called as *cell*, and the structure of the neural networks cannot be largely modified due to the structure of the *cell*. This limited structure design leads to a limited expression of the neural architecture and sub-optimal solutions. Another limitation is that computational cost for NAS is really high to simulate all possible architectures, which makes it difficult in practice.

8.3 Future Directions

Our works in this thesis open some challenging research directions. Below, we concisely describe the future work that potentially contribute to advance adaptation in DNNs:

- **Cross-domain few-shot learning.** There are some various methods to tackle the problem [Triantafillou et al., 2020]. However, The Meta-dataset [Triantafillou et al., 2020]

contains visual data of various datasets with overlapped classes and various domains. This is a very challenging dataset because the number of shot can vary from one episode to another episode, and the proposed algorithms need to adapt with such changes when predicting query images.

- **Multi-source incremental learning.** Current incremental learning algorithms are capable of alleviating *catastrophic forgetting* in a standard setup within an environment. A multi-source incremental learning from various datasets is a more realistic scenario. The ultimate goal of this problem is to incrementally improve the network without any boundary for incoming tasks including new domains, new concepts, and new inputs. The visual domain decathlon dataset [Rebuffi et al., 2017a] can be a candidate for visual task recognition.

- **Open-ended reinforcement learning.** In reinforcement learning, intelligent agents need to face uncertainties in acquiring new skills. A desired capability of the agents is an open-ended learning process such that the agents can learn with dynamically task distributions and training objectives. The environments generated by a simulator proposed in [Team et al., 2021] are developed specifically to assess the long-life learning capability of an agent.

- **Multi-agent incremental learning.** In the current incremental learning scenario, the problem is limited to human supervision or the designed rewards. In a cooperative situation, an agent interacts with other agents to gain more knowledge about the others. Specifically, the model for each network evolves based on the context and interaction with other agents.

- **Vision and language model adaptation.** A joint model for multi-modalities is underexplored for the case of sequential learning. Our brain obtains information from multi-source and modalities, but the current machine learning model cannot mimic this capability to learn new skills/tasks.

- **incremental semantic segmentation.** In computer vision, semantic meaning might change from time to time. This problem is known as semantic drift [Lesort et al., 2021]. After deployment, we prefer to fine-tune the model only for the changes affected by this semantic change rather than retraining the model from scratch.

- **Knowledge reasoning.** How the body of knowledge of similar objects can help to understand the unseen object with less effort. This is also related to the problem of causal learning.

Bibliography

ACHILLE, A. AND SOATTO, S., 2018. Emergence of invariance and disentanglement in deep representations. *The Journal of Machine Learning Research*, 19, 1 (2018), 1947–1980. (cited on page 100)

AL-SHEDIVAT, M.; BANSAL, T.; BURDA, Y.; SUTSKEVER, I.; MORDATCH, I.; AND ABBEEL, P., 2018. Continuous adaptation via meta-learning in nonstationary and competitive environments. In *International Conference on Learning Representations*. (cited on page 63)

ALEMI, A. A.; FISCHER, I.; DILLON, J. V.; AND MURPHY, K., 2017. Deep variational information bottleneck. In *International conference on Learning Representations*. (cited on page 101)

ALFASSY, A.; KARLINSKY, L.; AIDES, A.; SHTOK, J.; HARARY, S.; FERIS, R.; GIRYES, R.; AND BRONSTEIN, A. M., 2019. Laso: Label-set operations networks for multi-label few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6548–6557. (cited on pages xxi, 46, 57, and 60)

ANDRYCHOWICZ, M.; DENIL, M.; GOMEZ, S.; HOFFMAN, M. W.; PFAU, D.; SCHAUL, T.; SHILLINGFORD, B.; AND DE FREITAS, N., 2016. Learning to learn by gradient descent by gradient descent. In *Advances in neural information processing systems*. (cited on page 16)

ANTONIOU, A.; EDWARDS, H.; AND STORKEY, A., 2019. How to train your maml. In *International Conference on Learning Representations*. (cited on pages 17, 63, and 66)

ANTONIOU, A. AND STORKEY, A. J., 2019. Learning to learn by self-critique. In *Advances in Neural Information Processing Systems*, 9936–9946. (cited on page 71)

ARAZO, E.; ORTEGO, D.; ALBERT, P.; O'CONNOR, N.; AND MCGUINNESS, K., 2019. Unsupervised label noise modeling and loss correction. In *International Conference on Machine Learning*, 312–321. (cited on pages 21 and 96)

BAKER, B.; GUPTA, O.; NAIK, N.; AND RASKAR, R., 2016. Designing neural network architectures using reinforcement learning. In *International Conference on Learning Representations*. (cited on page 21)

BALCAN, M.-F.; KHODAK, M.; AND TALWALKAR, A., 2019. Provable guarantees for gradient-based meta-learning. In *International Conference on Machine Learning*, 424–433. (cited on page 16)

BARUTCUOGLU, Z.; SCHAPIRE, R. E.; AND TROYANSKAYA, O. G., 2006. Hierarchical multi-label prediction of gene function. *Bioinformatics*, 22, 7 (2006), 830–836. (cited on page 45)

BASRI, R. AND JACOBS, D. W., 2003. Lambertian reflectance and linear subspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25 (2003), 218–233. (cited on pages 8 and 24)

BELOUADAH, E. AND POPESCU, A., 2019. Il2m: Class incremental learning with dual memory. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 583–592. (cited on page 20)

BENGIO, Y., 2000. Gradient-based optimization of hyperparameters. *Neural computation*, 12, 8 (2000), 1889–1900. (cited on page 98)

BERTINETTO, L.; HENRIQUES, J. F.; TORR, P.; AND VEDALDI, A., 2019. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations*. (cited on page 71)

BOMMASANI, R.; HUDSON, D. A.; ADELI, E.; ALTMAN, R.; ARORA, S.; VON ARX, S.; BERNSTEIN, M. S.; BOHG, J.; BOSSELUT, A.; BRUNSKILL, E.; BRYNJOLFSSON, E.; BUCH, S.; CARD, D.; CASTELLON, R.; CHATTERJI, N.; CHEN, A.; CREEL, K.; DAVIS, J. Q.; DEMSZKY, D.; DONAHUE, C.; DOUMBOUYA, M.; DURMUS, E.; ERMON, S.; ETCHEMENDY, J.; ETHAYARAJH, K.; FEI-FEI, L.; FINN, C.; GALE, T.; GILLESPIE, L.; GOEL, K.; GOODMAN, N.; GROSSMAN, S.; GUHA, N.; HASHIMOTO, T.; HENDERSON, P.; HEWITT, J.; HO, D. E.; HONG, J.; HSU, K.; HUANG, J.; ICARD, T.; JAIN, S.; JURAFSKY, D.; KALLURI, P.; KARAMCHETI, S.; KEELING, G.; KHANI, F.; KHATTAB, O.; KOH, P. W.; KRASS, M.; KRISHNA, R.; KUDITIPUDI, R.; KUMAR, A.; LADHAK, F.; LEE, M.; LEE, T.; LESKOVEC, J.; LEVENT, I.; LI, X. L.; LI, X.; MA, T.; MALIK, A.; MANNING, C. D.; MIRCHANDANI, S.; MITCHELL, E.; MUNYIKWA, Z.; NAIR, S.; NARAYAN, A.; NARAYANAN, D.; NEWMAN, B.; NIE, A.; NIEBLES, J. C.; NILFOROSHAN, H.; NYARKO, J.; OGUT, G.; ORR, L.; PAPADIMITRIOU, I.; PARK, J. S.; PIECH, C.; PORTELANCE, E.; POTTS, C.; RAGHUNATHAN, A.; REICH, R.; REN, H.; RONG, F.; ROOHANI, Y.; RUIZ, C.; RYAN, J.; RÉ, C.; SADIGH, D.; SAGAWA, S.; SANTHANAM, K.; SHIH, A.; SRINIVASAN, K.; TAMKIN, A.; TAORI, R.; THOMAS, A. W.; TRAMÈR, F.; WANG, R. E.; WANG, W.; WU, B.; WU, J.; WU, Y.; XIE, S. M.; YASUNAGA, M.; YOU, J.; ZAHARIA, M.; ZHANG, M.; ZHANG, T.; ZHANG, X.; ZHANG, Y.; ZHENG, L.; ZHOU, K.; AND LIANG, P., 2021. On the opportunities and risks of foundation models. (cited on page 4)

CARON, M.; TOUVRON, H.; MISRA, I.; JÉGOU, H.; MAIRAL, J.; BOJANOWSKI, P.; AND JOULIN, A., 2021. Emerging properties in self-supervised vision transformers. *arXiv preprint arXiv:2104.14294*, (2021). (cited on page 7)

CASTRO, F. M.; MARÍN-JIMÉNEZ, M. J.; GUIL, N.; SCHMID, C.; AND ALAHARI, K., 2018. End-to-end incremental learning. In *Proceedings of the European conference on computer vision (ECCV)*, 233–248. (cited on page 20)

CESA-BIANCHI, N.; RE, M.; AND VALENTINI, G., 2012. Synergy of multi-label hierarchical ensembles, data fusion, and cost-sensitive methods for gene functional inference. *Machine Learning*, 88, 1-2 (2012), 209–241. (cited on page 45)

CHALUP, S. K., 2002. Incremental learning in biological and machine learning systems. *International Journal of Neural Systems*, 12, 06 (2002), 447–465. (cited on page 82)

CHAUDHRY, A.; DOKANIA, P. K.; AJANTHAN, T.; AND TORR, P. H., 2018. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 532–547. (cited on page 20)

CHEN, G.; CHOI, W.; YU, X.; HAN, T.; AND CHANDRAKER, M., 2017. Learning efficient object detection models with knowledge distillation. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 742–751. (cited on page 7)

CHEN, P. AND SUTER, D., 2006. An analysis of linear subspace approaches for computer vision and pattern recognition. *International Journal of Computer Vision*, 68, 1 (2006), 83–106. (cited on page 33)

CHEN, W.-Y.; LIU, Y.-C.; KIRA, Z.; WANG, Y.-C. F.; AND HUANG, J.-B., 2019a. A closer look at few-shot classification. In *International Conference on Learning Representations*. (cited on pages 36, 38, 41, and 70)

CHEN, W.-Y.; LIU, Y.-C.; KIRA, Z.; WANG, Y.-C. F.; AND HUANG, J.-B., 2019b. A closer look at few-shot classification. In *International Conference on Learning Representations*. (cited on page 88)

CHEN, Y.; YANG, T.; ZHANG, X.; MENG, G.; XIAO, X.; AND SUN, J., 2019c. Detnas: Backbone search for object detection. In *Advances in Neural Information Processing Systems*, 6638–6648. (cited on page 21)

CHERAGHIAN, A.; RAHMAN, S.; FANG, P.; ROY, S. K.; PETERSSON, L.; AND HARANDI, M., 2021. Semantic-aware knowledge distillation for few-shot class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on page 20)

CHERIAN, A., 2013. Suvrit sra, arindam banerjee, and nikolaos papanikolopoulos. jensen-bregman logdet divergence with application to efficient similarity search for covariance matrices. *IEEE transactions on pattern analysis and machine intelligence*, 35, 9 (2013), 2161–2174. (cited on page 28)

CUNNINGHAM, J. P. AND GHAHRAMANI, Z., 2015. Linear dimensionality reduction: Survey, insights, and generalizations. *The Journal of Machine Learning Research*, 16, 1 (2015), 2859–2900. (cited on page 27)

DESJARDINS, G.; SIMONYAN, K.; PASCANU, R.; ET AL., 2015. Natural neural networks. In *Advances in Neural Information Processing Systems*, 2071–2079. (cited on page 68)

DEVLIN, J.; CHANG, M.-W.; LEE, K.; AND TOUTANOVA, K., 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. (cited on page 2)

DONG, X. AND YANG, Y., 2019. Searching for a robust neural architecture in four gpu hours. In *The IEEE Conference on Computer Vision and Pattern Recognition*. (cited on page 95)

DOSOVITSKIY, A.; BEYER, L.; KOLESNIKOV, A.; WEISSENBORN, D.; ZHAI, X.; UNTERTHINER, T.; DEHGHANI, M.; MINDERER, M.; HEIGOLD, G.; GELLY, S.; USZKOREIT, J.; AND HOULSBY, N., 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*. https://openreview.net/forum?id=YicbFdNTTy. (cited on page 2)

EDELMAN, A.; ARIAS, T. A.; AND SMITH, S. T., 1998. The geometry of algorithms with orthogonality constraints. *SIAM journal on Matrix Analysis and Applications*, 20 (1998), 303–353. (cited on page 28)

ELSKEN, T.; METZEN, J. H.; AND HUTTER, F., 2019. Efficient multi-objective neural architecture search via lamarckian evolution. In *International Conference on Learning Representations*. (cited on pages 21 and 95)

ELSKEN, T.; STAFFLER, B.; METZEN, J. H.; AND HUTTER, F., 2020. Meta-learning of neural architectures for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12365–12375. (cited on page 7)

FANG, P.; ZHOU, J.; ROY, S. K.; PETERSSON, L.; AND HARANDI, M., 2019. Bilinear attention networks for person retrieval. In *Proceedings of the IEEE International Conference on Computer Vision*, 8030–8039. (cited on page 15)

FEI-FEI, L.; FERGUS, R.; AND PERONA, P., 2003. A bayesian approach to unsupervised one-shot learning of object categories. In *Proceedings of IEEE International Conference on Computer Vision*, vol. 2, 1134–1141. (cited on page 14)

FELDMAN, J., 1997. The structure of perceptual categories. *Journal of mathematical psychology*, 41, 2 (1997), 145–170. (cited on pages xv, 13, and 14)

FINN, C.; ABBEEL, P.; AND LEVINE, S., 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of International Conference on Machine Learning*, 1126–1135. (cited on pages xvi, 6, 16, 17, 23, 63, 65, 66, 69, 70, 71, and 72)

FLENNERHAG, S.; MORENO, P. G.; LAWRENCE, N.; AND DAMIANOU, A., 2019. Transferring knowledge across learning processes. In *International Conference on Learning Representations*. (cited on page 69)

FLENNERHAG, S.; RUSU, A. A.; PASCANU, R.; VISIN, F.; YIN, H.; AND HADSELL, R., 2020. Meta-learning with warped gradient descent. In *International Conference on Learning Representations*. (cited on pages 17 and 23)

FRENCH, R. M., 1999. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3, 4 (1999), 128–135. (cited on page 81)

FUKUNAGA, K. AND KOONTZ, W. L. G., 1970. Application of the karhunen-loève expansion to feature selection and ordering. *IEEE Trans. Comput.*, 19, 4 (1970), 311–318. (cited on page 27)

FUSI, S.; DREW, P. J.; AND ABBOTT, L. F., 2005. Cascade models of synaptically stored memories. *Neuron*, 45, 4 (2005), 599–611. (cited on page 19)

GANIN, Y. AND LEMPITSKY, V., 2015. Unsupervised domain adaptation by backpropagation. In *Proceedings of the 32nd International Conference on Machine Learning*, vol. 37 of *Proceedings of Machine Learning Research*, 1180–1189. (cited on page 5)

GARCIA, V. AND BRUNA, J., 2018. Few-shot learning with graph neural networks. In *International Conference on Learning Representations*. (cited on page 16)

GARNELO, M.; SCHWARZ, J.; ROSENBAUM, D.; VIOLA, F.; REZENDE, D. J.; ESLAMI, S.; AND TEH, Y. W., 2018. Neural processes. *arXiv preprint arXiv:1807.01622*, (2018). (cited on pages 71 and 72)

GEPPERTH, A. AND KARAOGUZ, C., 2016. A bio-inspired incremental learning architecture for applied perceptual problems. *Cognitive Computation*, 8, 5 (2016), 924–934. (cited on page 82)

GIDARIS, S. AND KOMODAKIS, N., 2018a. Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4367–4375. (cited on pages 5, 15, and 88)

GIDARIS, S. AND KOMODAKIS, N., 2018b. Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4367–4375. (cited on pages 26 and 34)

GIDARIS, S. AND KOMODAKIS, N., 2019. Generating classification weights with GNN denoising autoencoders for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. (cited on pages 34 and 71)

GONG, X.; CHANG, S.; JIANG, Y.; AND WANG, Z., 2019. Autogan: Neural architecture search for generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 3224–3234. (cited on page 21)

GRANT, E.; FINN, C.; LEVINE, S.; DARRELL, T.; AND GRIFFITHS, T., 2018. Recasting gradient-based meta-learning as hierarchical bayes. In *International Conference on Learning Representations*. (cited on page 16)

GREEN, C. AND BAVELIER, D., 2009. Exercising your brain: A review of human brain plasticity and training-induced learning. *Psychology and aging*, 23 (01 2009), 692–701. (cited on page 1)

GUO, S.; HUANG, W.; ZHANG, H.; ZHUANG, C.; DONG, D.; SCOTT, M. R.; AND HUANG, D., 2018. Curriculumnet: Weakly supervised learning from large-scale web images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 135–150. (cited on page 22)

GUO, S.; HUANG, W.; ZHANG, X.; SRIKHANTA, P.; CUI, Y.; LI, Y.; R.SCOTT, M.; ADAM, H.; AND BELONGIE, S., 2019. The imaterialist fashion attribute dataset. *arXiv preprint arXiv:1906.05750*, (2019). (cited on pages 46 and 58)

HAMM, J. AND LEE, D. D., 2008. Grassmann discriminant analysis: a unifying view on subspace-based learning. In *Proceedings of the 25th international conference on Machine learning*, 376–383. (cited on page 29)

HAN, B.; YAO, Q.; YU, X.; NIU, G.; XU, M.; HU, W.; TSANG, I.; AND SUGIYAMA, M., 2018. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in neural information processing systems*, 8527–8537. (cited on pages 20, 22, 96, 104, 105, 106, 107, and 108)

HARANDI, M.; HARTLEY, R.; SALZMANN, M.; AND TRUMPF, J., 2016. *Dictionary Learning on Grassmann Manifolds*, 145–172. (cited on page 8)

HARANDI, M.; HARTLEY, R.; SHEN, C.; LOVELL, B.; AND SANDERSON, C., 2015. Extrinsic methods for coding and dictionary learning on grassmann manifolds. 114, 2–3 (2015), 113–136. (cited on page 8)

HARANDI, M. T.; SALZMANN, M.; AND HARTLEY, R., 2014. From manifold to manifold: Geometry-aware dimensionality reduction for spd matrices. In *European conference on computer vision*, 17–32. (cited on pages 8 and 28)

HE, K.; FAN, H.; WU, Y.; XIE, S.; AND GIRSHICK, R., 2019. Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722*, (2019). (cited on page 7)

HE, K.; ZHANG, X.; REN, S.; AND SUN, J., 2016. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*. (cited on pages xv, 2, 3, 5, 70, 89, and 105)

HEIDER, D.; SENGE, R.; CHENG, W.; AND HÜLLERMEIER, E., 2013. Multilabel classification for exploiting cross-resistance information in hiv-1 drug resistance prediction. *Bioinformatics*, 29, 16 (2013), 1946–1952. (cited on page 45)

HILLIARD, N.; PHILLIPS, L.; HOWLAND, S.; YANKOV, A.; CORLEY, C. D.; AND HODAS, N. O., 2018. Few-shot learning with metric-agnostic conditional embeddings. *arXiv preprint arXiv:1802.04376*, (2018). (cited on page 36)

HINTON, G.; VINYALS, O.; AND DEAN, J., 2015. Distilling the knowledge in a neural network. (cited on pages 7 and 20)

HOSPEDALES, T. M.; ANTONIOU, A.; MICAELLI, P.; AND STORKEY, A. J., 2020. Meta-learning in neural networks: A survey. *CoRR*, abs/2004.05439 (2020). https://arxiv.org/abs/2004.05439. (cited on page 16)

HOU, R.; CHANG, H.; BINGPENG, M.; SHAN, S.; AND CHEN, X., 2019a. Cross attention network for few-shot classification. In *Advances in Neural Information Processing Systems*, 4005–4016. (cited on page 16)

HOU, S.; PAN, X.; LOY, C. C.; WANG, Z.; AND LIN, D., 2019b. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on pages xix, 8, 19, 20, 82, 83, 84, 87, 88, 89, 90, 91, 92, 93, and 94)

INOUE, N.; SIMO-SERRA, E.; YAMASAKI, T.; AND ISHIKAWA, H., 2017. Multi-label fashion image classification with minimal human supervision. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2261–2267. (cited on page 45)

ISCEN, A.; ZHANG, J.; LAZEBNIK, S.; AND SCHMID, C., 2020. Memory-efficient incremental learning through feature adaptation. *arXiv preprint arXiv:2004.00713*, (2020). (cited on page 19)

JIANG, L.; ZHOU, Z.; LEUNG, T.; LI, L.-J.; AND FEI-FEI, L., 2018. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International Conference on Machine Learning*, 2304–2313. (cited on pages 21, 96, 105, 106, and 107)

KEMKER, R. AND KANAN, C., 2018. Fearnet: Brain-inspired model for incremental learning. In *International Conference on Learning Representations*. (cited on page 19)

KIM, J.; KIM, H.; AND KIM, G., 2020. Model-agnostic boundary-adversarial sampling for test-time generalization in few-shot learning. In *European Conference on Computer Vision (ECCV)*. (cited on page 34)

KINGMA, D. P. AND BA, J. L., 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*. (cited on pages 5, 59, and 70)

KINGMA, D. P. AND WELLING, M., 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, (2013). (cited on pages 100 and 101)

KIRKPATRICK, J.; PASCANU, R.; RABINOWITZ, N.; VENESS, J.; DESJARDINS, G.; RUSU, A. A.; MILAN, K.; QUAN, J.; RAMALHO, T.; GRABSKA-BARWINSKA, A.; ET AL., 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114, 13 (2017), 3521–3526. (cited on pages 19 and 81)

KOCH, G.; ZEMEL, R.; AND SALAKHUTDINOV, R., 2015. Siamese neural networks for one-shot image recognition. In *Proceedings of International Conference on Machine Learning Deep Learning 2015 Workshop*. (cited on pages 15 and 23)

KONIUSZ, P.; CHERIAN, A.; AND PORIKLI, F., 2016. Tensor representations via kernel linearization for action recognition from 3d skeletons. In *European conference on computer vision*, 37–53. (cited on page 28)

KONIUSZ, P.; TAS, Y.; ZHANG, H.; HARANDI, M.; PORIKLI, F.; AND ZHANG, R., 2018. Museum exhibit identification challenge for the supervised domain adaptation and beyond. In *The European Conference on Computer Vision*. (cited on pages 34, 36, 46, and 59)

KONIUSZ, P. AND ZHANG, H., 2020a. Power normalizations in fine-grained image, few-shot image and graph classification. *TPAMI*, (2020). (cited on page 17)

KONIUSZ, P. AND ZHANG, H., 2020b. Power normalizations in fine-grained image, few-shot image and graph classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2020). (cited on pages 32, 33, and 42)

KRIZHEVSKY, A., 2009. Learning multiple layers of features from tiny images. Technical report. (cited on page 89)

KRIZHEVSKY, A.; SUTSKEVER, I.; AND HINTON, G. E., 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. (cited on pages xv, 3, 4, 5, 70, and 114)

KRIZHEVSKY, A.; SUTSKEVER, I.; AND HINTON, G. E., 2017. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60, 6 (May 2017), 84–90. (cited on page 2)

KRIZHEVSKY, A. ET AL., 2009. Learning multiple layers of features from tiny images. Technical report, Citeseer. (cited on pages 34 and 36)

KURAKIN, A.; GOODFELLOW, I.; AND BENGIO, S., 2017. Adversarial machine learning at scale. (cited on page 2)

LAKE, B. M.; SALAKHUTDINOV, R.; AND TENENBAUM, J. B., 2015a. Human-level concept learning through probabilistic program induction. *Science*, 350, 6266 (2015), 1332–1338. (cited on pages xv, 3, and 69)

LAKE, B. M.; SALAKHUTDINOV, R.; AND TENENBAUM, J. B., 2015b. Human-level concept learning through probabilistic program induction. *Science*, 350 (Dec. 2015), 1332–1338. (cited on page 3)

LARSEN, J.; HANSEN, L. K.; SVARER, C.; AND OHLSSON, M., 1996. Design and regularization of neural networks: the optimal use of a validation set. In *Neural Networks for Signal Processing VI. Proceedings of the 1996 IEEE Signal Processing Society Workshop*, 62–71. (cited on page 98)

LEE, K.; MAJI, S.; RAVICHANDRAN, A.; AND SOATTO, S., 2019a. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 10657–10665. (cited on pages 34 and 35)

LEE, K.; MAJI, S.; RAVICHANDRAN, A.; AND SOATTO, S., 2019b. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 10657–10665. (cited on page 70)

LEE, Y. AND CHOI, S., 2018. Gradient-based meta-learning with learned layerwise metric and subspace. In *International Conference on Machine Learning*, 2933–2942. (cited on pages 68, 71, and 72)

LEGG, S. AND HUTTER, M., 2007. Universal intelligence: A definition of machine intelligence. *Minds and Machines*, 17, 4 (2007). (cited on page 81)

LESORT, T.; CACCIA, M.; AND RISH, I., 2021. Understanding continual learning settings with data distribution drift analysis. (cited on page 115)

LI, H.; EIGEN, D.; DODGE, S.; ZEILER, M.; AND WANG, X., 2019a. Finding task-relevant features for few-shot learning by category traversal. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1–10. (cited on page 34)

LI, R.; JIAO, Q.; CAO, W.; WONG, H.-S.; AND WU, S., 2020. Model adaptation: Unsupervised domain adaptation without source data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on page 5)

LI, W.; WANG, L.; XU, J.; HUO, J.; YANG, G.; AND LUO, J., 2019b. Revisiting local descriptor based image-to-class measure for few-shot learning. In *CVPR*. (cited on page 41)

LI, X.; ZHOU, Y.; WU, T.; SOCHER, R.; AND XIONG, C., 2019c. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *International Conference on Machine Learning*, 3925–3934. (cited on page 7)

LI, Z. AND HOIEM, D., 2017. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40, 12 (2017), 2935–2947. (cited on pages xix, 20, 81, 82, 83, 84, 90, 91, 92, 93, and 94)

LI, Z.; ZHOU, F.; CHEN, F.; AND LI, H., 2017. Meta-sgd: Learning to learn quickly for few shot learning. *arXiv preprint arXiv:1707.09835*, (2017). (cited on pages xvi, 17, 64, 66, and 71)

LIAN, D.; ZHENG, Y.; XU, Y.; LU, Y.; LIN, L.; ZHAO, P.; HUANG, J.; AND GAO, S., 2020. Towards fast adaptation of neural architectures with meta learning. In *International Conference on Learning Representations*. (cited on pages 21 and 98)

LIFCHITZ, Y.; AVRITHIS, Y.; PICARD, S.; AND BURSUC, A., 2019. Dense classification and implanting for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 9258–9267. (cited on page 15)

LIN, T.-Y.; MAIRE, M.; BELONGIE, S.; HAYS, J.; PERONA, P.; RAMANAN, D.; DOLLÁR, P.; AND ZITNICK, C. L., 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, 740–755. (cited on pages 46 and 56)

LIN, T.-Y. AND MAJI, S., 2017. Improved Bilinear Pooling with CNNs. *British Machine Vision Conference*, (2017). (cited on page 42)

LIU, C.; CHEN, L.-C.; SCHROFF, F.; ADAM, H.; HUA, W.; YUILLE, A. L.; AND FEI-FEI, L., 2019a. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 82–92. (cited on pages 21 and 98)

LIU, H.; SIMONYAN, K.; AND YANG, Y., 2019b. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*. (cited on pages 7, 21, 95, 97, 98, and 104)

LIU, Y.; CHEN, K.; LIU, C.; QIN, Z.; LUO, Z.; AND WANG, J., 2019c. Structured knowledge distillation for semantic segmentation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2599–2608. (cited on page 7)

LIU, Y.; LEE, J.; PARK, M.; KIM, S.; YANG, E.; HWANG, S.; AND YANG, Y., 2019d. Learning to propagate labels: Transductive propagation network for few-shot learning. In *International Conference on Learning Representations*. (cited on page 16)

LIU, Y.; SCHIELE, B.; AND SUN, Q., 2020a. An ensemble of epoch-wise empirical bayes for few-shot learning. In *European Conference on Computer Vision (ECCV)*. (cited on pages 23 and 34)

LIU, Y.; SU, Y.; LIU, A.-A.; SCHIELE, B.; AND SUN, Q., 2020b. Mnemonics training: Multi-class incremental learning without forgetting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12245–12254. (cited on pages 19, 84, 87, 89, 90, and 91)

LIU, Z.; LUO, P.; WANG, X.; AND TANG, X., 2015. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision*. (cited on page 71)

LOPEZ-PAZ, D. AND RANZATO, M., 2017. Gradient episodic memory for continual learning. In *Advances in neural information processing systems*, 6467–6476. (cited on pages 20 and 82)

LUO, R.; TIAN, F.; QIN, T.; CHEN, E.; AND LIU, T.-Y., 2018. Neural architecture optimization. In *Advances in neural information processing systems*, 7816–7827. (cited on page 95)

MA, R.; FANG, P.; DRUMMOND, T.; AND HARANDI, M., 2021. Adaptive poincaré point to set distance for few-shot classification. *CoRR*, abs/2112.01719 (2021). (cited on page 23)

MAJI, S.; BERG, A. C.; AND MALIK, J., 2012. Efficient classification for additive kernel svms. *IEEE transactions on pattern analysis and machine intelligence*, 35, 1 (2012), 66–77. (cited on page 28)

MALACH, E. AND SHALEV-SHWARTZ, S., 2017. Decoupling" when to update" from" how to update". In *Advances in Neural Information Processing Systems*, 960–970. (cited on pages 105, 106, 107, and 108)

MALLYA, A.; DAVIS, D.; AND LAZEBNIK, S., 2018. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European Conference on Computer Vision*. (cited on page 20)

MALLYA, A. AND LAZEBNIK, S., 2018. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7765–7773. (cited on page 20)

MASANA, M.; LIU, X.; TWARDOWSKI, B.; MENTA, M.; BAGDANOV, A. D.; AND VAN DE WEIJER, J., 2020. Class-incremental learning: survey and performance evaluation. *CoRR*, abs/2010.15277 (2020). (cited on pages xvi and 18)

MCCLELLAND, J. L.; MCNAUGHTON, B. L.; AND O'REILLY, R. C., 1995. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102, 3 (1995), 419. (cited on pages 18 and 81)

MCCLOSKEY, M. AND COHEN, N. J., 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, vol. 24, 109–165. Elsevier. (cited on pages 18 and 81)

MENON, A. K. AND ELKAN, C., 2011. Fast algorithms for approximating the singular value decomposition. *ACM Trans. Knowl. Discov. Data*, 5 (2011), 13:1–13:36. (cited on page 42)

MENSINK, T.; VERBEEK, J.; PERRONNIN, F.; AND CSURKA, G., 2012. Metric learning for large scale image classification: Generalizing to new classes at near-zero cost. In *European Conference on Computer Vision*, 488–501. (cited on page 88)

MILLER, E. G.; MATSAKIS, N. E.; AND VIOLA, P. A., 2000. Learning from one example through shared densities on transforms. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 464–471. (cited on page 14)

MITCHELL, T. M., 1997. *Machine Learning*. McGraw-Hill, New York. (cited on page 1)

MNIH, V.; KAVUKCUOGLU, K.; SILVER, D.; RUSU, A. A.; VENESS, J.; BELLEMARE, M. G.; GRAVES, A.; RIEDMILLER, M.; FIDJELAND, A. K.; OSTROVSKI, G.; ET AL., 2015. Human-level control through deep reinforcement learning. *nature*, 518, 7540 (2015), 529–533. (cited on page 2)

MURASE, H. AND NAYAR, S. K., 1997. Detection of 3d objects in cluttered scenes using hierarchical eigenspace. *Pattern recognition letters*, 18, 4 (1997), 375–384. (cited on page 27)

NATARAJAN, N.; DHILLON, I. S.; RAVIKUMAR, P. K.; AND TEWARI, A., 2013. Learning with noisy labels. In *Advances in neural information processing systems*, 1196–1204. (cited on page 96)

NICHOL, A.; ACHIAM, J.; AND SCHULMAN, J., 2018. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, (2018). (cited on pages 17, 69, and 71)

OQUAB, M.; BOTTOU, L.; LAPTEV, I.; AND SIVIC, J., 2014. Learning and transferring mid-level image representations using convolutional neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 1717–1724. (cited on page 5)

ORESHKIN, B.; RODRÍGUEZ LÓPEZ, P.; AND LACOSTE, A., 2018. Tadam: Task dependent adaptive metric for improved few-shot learning. In *Advances in Neural Information Processing Systems*, 719–728. (cited on pages 34, 35, and 36)

PACHET, F. AND ROY, P., 2009. Improving multilabel analysis of music titles: A large-scale validation of the correction approach. *IEEE Transactions on Audio, Speech, and Language Processing*, 17, 2 (2009), 335–343. (cited on page 45)

PAN, S. J. AND YANG, Q., 2009. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22, 10 (2009), 1345–1359. (cited on page 4)

PAPERNOT, N.; ABADI, M.; ERLINGSSON, Ú.; GOODFELLOW, I. J.; AND TALWAR, K., 2017. Semi-supervised knowledge transfer for deep learning from private training data. In *5th International Conference on Learning Representations*. (cited on page 7)

PARK, E. AND OLIVA, J. B., 2019. Meta-curvature. In *Advances in Neural Information Processing Systems*, 3309–3319. (cited on pages 17, 23, 66, and 71)

PASZKE, A.; GROSS, S.; CHINTALA, S.; CHANAN, G.; YANG, E.; DEVITO, Z.; LIN, Z.; DESMAISON, A.; ANTIGA, L.; AND LERER, A., 2017. Automatic differentiation in pytorch. In *NIPS Autodiff Workshop*. (cited on pages 28 and 89)

PATRINI, G.; ROZZA, A.; KRISHNA MENON, A.; NOCK, R.; AND QU, L., 2017. Making deep neural networks robust to label noise: A loss correction approach. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1944–1952. (cited on pages 20, 21, 96, 105, 106, 107, and 108)

PEREZ, E.; STRUB, F.; DE VRIES, H.; DUMOULIN, V.; AND COURVILLE, A., 2018. Film: Visual reasoning with a general conditioning layer. In *Thirty-Second AAAI Conference on Artificial Intelligence*. (cited on page 17)

PHAM, H.; GUAN, M. Y.; ZOPH, B.; LE, Q. V.; AND DEAN, J., 2018. Efficient neural architecture search via parameter sharing. In *ICML*, 4092–4101. (cited on pages 7 and 95)

QI, G.-J.; HUA, X.-S.; RUI, Y.; TANG, J.; MEI, T.; AND ZHANG, H.-J., 2007. Correlative multi-label video annotation. In *Proceedings of the ACM international conference on Multimedia*, 17–26. (cited on page 45)

QIAO, S.; LIU, C.; SHEN, W.; AND YUILLE, A. L., 2018. Few-shot image recognition by predicting parameters from activations. In *IEEE Conference on Computer Vision and Pattern Recognition*. (cited on pages xv, 5, 15, 26, 70, and 71)

RAGHU, A.; RAGHU, M.; BENGIO, S.; AND VINYALS, O., 2020. Rapid learning or feature reuse? towards understanding the effectiveness of maml. In *International Conference on Learning Representations*. (cited on page 70)

RAGHU, M.; GILMER, J.; YOSINSKI, J.; AND SOHL-DICKSTEIN, J., 2017. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *Advances in Neural Information Processing Systems*, 6076–6085. (cited on page 94)

RAJASEGARAN, J.; HAYAT, M.; KHAN, S. H.; KHAN, F. S.; AND SHAO, L., 2019. Random path selection for continual learning. In *Advances in Neural Information Processing Systems 32*, 12669–12679. (cited on pages 7 and 20)

RAJESWARAN, A.; FINN, C.; KAKADE, S.; AND LEVINE, S., 2019. Meta-learning with implicit gradients. In *Advances in Neural Information Processing Systems*. (cited on page 63)

RASCH, B. AND BORN, J., 2013. About sleep's role in memory. *Physiological reviews*, (2013). (cited on pages 19 and 82)

RAVI, S. AND LAROCHELLE, H., 2017. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*. (cited on pages xxi, 14, 23, 34, 35, 63, 70, and 71)

RAVICHANDRAN, A.; BHOTIKA, R.; AND SOATTO, S., 2019. Few-shot learning with embedded class models and shot-free meta training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 331–339. (cited on page 34)

REAL, E.; AGGARWAL, A.; HUANG, Y.; AND LE, Q. V., 2019. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, vol. 33, 4780–4789. (cited on pages 21, 95, 98, and 104)

REAL, E.; MOORE, S.; SELLE, A.; SAXENA, S.; SUEMATSU, Y. L.; TAN, J.; LE, Q. V.; AND KURAKIN, A., 2017. Large-scale evolution of image classifiers. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, 2902–2911. (cited on pages 21 and 95)

REBUFFI, S.-A.; BILEN, H.; AND VEDALDI, A., 2017a. Learning multiple visual domains with residual adapters. In *NIPS*. (cited on page 115)

REBUFFI, S.-A.; KOLESNIKOV, A.; SPERL, G.; AND LAMPERT, C. H., 2017b. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2001–2010. (cited on pages 19, 81, 82, 84, 88, 90, and 91)

REED, S.; LEE, H.; ANGUELOV, D.; SZEGEDY, C.; ERHAN, D.; AND RABINOVICH, A., 2015. Training deep neural networks on noisy labels with bootstrapping. In *International Conference on Learning Representations*. (cited on page 21)

REN, M.; TRIANTAFILLOU, E.; RAVI, S.; SNELL, J.; SWERSKY, K.; TENENBAUM, J. B.; LAROCHELLE, H.; AND ZEMEL, R. S., 2018a. Meta-learning for semi-supervised few-shot classification. In *International Conference on Learning Representations*. (cited on page 34)

REN, M.; ZENG, W.; YANG, B.; AND URTASUN, R., 2018b. Learning to reweight examples for robust deep learning. In *International Conference on Machine Learning*, 4334–4343. (cited on pages 22 and 96)

RUSSAKOVSKY, O.; DENG, J.; SU, H.; ANDSANJEEV SATHEESH, J. K.; MA, S.; HUANG, Z.; KARPATHY, A.; KHOSLA, A.; AND MICHAEL BERNSTEIN, E. A., 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115 (2015), 211–252. (cited on page 35)

RUSU, A. A.; RABINOWITZ, N. C.; DESJARDINS, G.; SOYER, H.; KIRKPATRICK, J.; KAVUKCUOGLU, K.; PASCANU, R.; AND HADSELL, R., 2016. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, (2016). (cited on page 7)

RUSU, A. A.; RAO, D.; SYGNOWSKI, J.; VINYALS, O.; PASCANU, R.; OSINDERO, S.; AND HADSELL, R., 2019. Meta-learning with latent embedding optimization. In *International Conference on Learning Representations*. (cited on pages 17, 18, 23, 34, 63, 64, 66, 70, and 71)

SANTORO, A.; BARTUNOV, S.; BOTVINICK, M.; WIERSTRA, D.; AND LILLICRAP, T., 2016. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, 1842–1850. (cited on pages 15, 16, 45, and 65)

SAU, B. B. AND BALASUBRAMANIAN, V. N., 2016. Deep model compression: Distilling knowledge from noisy teachers. *ArXiv*, abs/1610.09650 (2016). (cited on page 7)

SAXENA, S. AND VERBEEK, J., 2016. Convolutional neural fabrics. In *Advances in Neural Information Processing Systems*, 4053–4061. (cited on page 95)

SELVARAJU, R. R.; COGSWELL, M.; DAS, A.; VEDANTAM, R.; PARIKH, D.; AND BATRA, D., 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. (cited on pages xvii, 40, and 41)

SHENG FOO, C.; DO, C. B.; AND NG, A. Y., 2008. Efficient multiple hyperparameter learning for log-linear models. In *Advances in Neural Information Processing Systems 20*, 377–384. (cited on page 98)

SHI, Y.; LIU, L.; YU, X.; AND LI, H., 2019. Spatial-aware feature aggregation for image based cross-view geo-localization. In *Advances in Neural Information Processing Systems*, 10090–10100. (cited on page 15)

SHIN, H.; LEE, J. K.; KIM, J.; AND KIM, J., 2017. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, 2990–2999. (cited on page 19)

SHWARTZ-ZIV, R. AND TISHBY, N., 2017. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, (2017). (cited on pages 103 and 107)

SIMON, C.; KONIUSZ, P.; NOCK, R.; AND HARANDI, M., 2020a. Adaptive subspaces for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. (cited on pages 5, 17, 23, and 36)

SIMON, C.; KONIUSZ, P.; NOCK, R.; AND HARANDI, M., 2020b. On modulating the gradient for meta-learning. In *European Conference on Computer Vision*, 556–572. (cited on page 23)

SNELL, J.; SWERSKY, K.; AND RICHARD, Z., 2017. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, 4077–4087. (cited on pages 15, 17, 24, 25, 26, 34, 35, 38, 39, 41, 42, 45, 55, and 70)

SRIVASTAVA, N.; HINTON, G.; KRIZHEVSKY, A.; SUTSKEVER, I.; AND SALAKHUTDINOV, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15, 1 (2014), 1929–1958. (cited on pages 7 and 20)

STANLEY, K. O.; CLUNE, J.; LEHMAN, J.; AND MIIKKULAINEN, R., 2019. Designing neural networks through neuroevolution. (2019), 24–35. (cited on pages 21 and 95)

SUN, Q.; LIU, Y.; CHUA, T.-S.; AND SCHIELE, B., 2019a. Meta-transfer learning for few-shot learning. In *The IEEE Conference on Computer Vision and Pattern Recognition*. (cited on pages 18, 63, and 71)

SUN, Q.; LIU, Y.; CHUA, T.-S.; AND SCHIELE, B., 2019b. Meta-transfer learning for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 403–412. (cited on page 35)

SUNG, F.; YANG, Y.; ZHANG, L.; XIANG, T.; TORR, P. H.; AND HOSPEDALES, T. M., 2018. Learning to compare: Relation network for few-shot learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1199–1208. (cited on pages 15, 23, 26, 35, 46, 55, and 61)

SUTSKEVER, I.; VINYALS, O.; AND LE, Q. V., 2014. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215 (2014). (cited on page 2)

TANAKA, D.; IKAMI, D.; YAMASAKI, T.; AND AIZAWA, K., 2018. Joint optimization framework for learning with noisy labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5552–5560. (cited on pages 21 and 105)

TANNO, R.; SAEEDI, A.; SANKARANARAYANAN, S.; ALEXANDER, D. C.; AND SILBERMAN, N., 2019. Learning from noisy labels by regularized estimation of annotator confusion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 11244–11253. (cited on page 96)

TARVAINEN, A. AND VALPOLA, H., 2017. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 1195–1204. (cited on page 7)

TEAM, O. E. L.; STOOKE, A.; MAHAJAN, A.; BARROS, C.; DECK, C.; BAUER, J.; SYGNOWSKI, J.; TREBACZ, M.; JADERBERG, M.; MATHIEU, M.; MCALEESE, N.; BRADLEY-SCHMIEG, N.; WONG, N.; PORCEL, N.; RAILEANU, R.; HUGHES-FITT, S.; DALIBARD, V.; AND CZARNECKI, W. M., 2021. Open-ended learning leads to generally capable agents. (cited on page 115)

THRUN, S., 1995. Is learning the n-th thing any easier than learning the first? In *Proceedings of the 8th International Conference on Neural Information Processing Systems*, 640–646. (cited on page 4)

TISHBY, N.; PEREIRA, F. C.; AND BIALEK, W., 2000. The information bottleneck method. *arXiv preprint physics/0004057*, (2000). (cited on page 99)

TODOROV, E.; EREZ, T.; AND TASSA, Y., 2012. Mujoco: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5026–5033. (cited on page 72)

TRIANTAFILLOU, E.; ZHU, T.; DUMOULIN, V.; LAMBLIN, P.; EVCI, U.; XU, K.; GOROSHIN, R.; GELADA, C.; SWERSKY, K.; MANZAGOL, P.-A.; AND LAROCHELLE, H., 2020. Meta-dataset: A dataset of datasets for learning to learn from few examples. In *International Conference on Learning Representations*. (cited on page 114)

TURK, M. A. AND PENTLAND, A. P., 1991a. Face recognition using eigenfaces. In *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 586–591. (cited on page 24)

TURK, M. A. AND PENTLAND, A. P., 1991b. Face recognition using eigenfaces. In *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 586–591. (cited on page 27)

VAHDAT, A., 2017. Toward robustness against label noise in training deep discriminative neural networks. In *Advances in Neural Information Processing Systems*, 5596–5605. (cited on pages 21 and 96)

VAN DE VEN, G. M. AND TOLIAS, A. S., 2019. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, (2019). (cited on pages 18 and 82)

VAN DER MAATEN, L. AND HINTON, G., 2008. *Journal of Machine Learning Research*, 9 (2008), 2579–2605. (cited on pages xv and 5)

VAN LOAN, C. F., 1976. Generalizing the singular value decomposition. *SIAM Journal on numerical Analysis*, 13, 1 (1976), 76–83. (cited on pages 85 and 87)

VENIAT, T. AND DENOYER, L., 2018. Learning time/memory-efficient deep architectures with budgeted super networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3492–3500. (cited on page 95)

VINYALS, O.; BLUNDELL, C.; LILLICRAP, T.; KAVUKCUOGLU, K.; AND WIERSTRA, D., 2016. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, 3630–3638. (cited on pages xxi, 14, 15, 16, 23, 26, 35, 38, 41, 45, 47, and 65)

WANG, J. AND CHERIAN, A., 2019. Gods: Generalized one-class discriminative subspaces for anomaly detection. In *The IEEE International Conference on Computer Vision*. (cited on page 24)

WANG, J.; YANG, Y.; MAO, J.; HUANG, Z.; HUANG, C.; AND XU, W., 2016. Cnn-rnn: A unified framework for multi-label image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2285–2294. (cited on page 48)

WANG, Y.; MA, X.; CHEN, Z.; LUO, Y.; YI, J.; AND BAILEY, J., 2019. Symmetric cross entropy for robust learning with noisy labels. In *The IEEE International Conference on Computer Vision (ICCV)*. (cited on page 96)

WANG, Y. AND YAO, Q., 2019. Few-shot learning: A survey. *CoRR*, abs/1904.05046 (2019). http://arxiv.org/abs/1904.05046. (cited on page 15)

WANG, Y.-X.; GIRSHICK, R.; HEBERT, M.; AND HARIHARAN, B., 2018. Low-shot learning from imaginary data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. (cited on page 26)

WANG, Z.; CHEN, T.; LI, G.; XU, R.; AND LIN, L., 2017. Multi-label image recognition by recurrently discovering attentional regions. In *Proceedings of the IEEE International Conference on Computer Vision*, 464–472. (cited on pages 48 and 55)

WEI, H.; FENG, L.; CHEN, X.; AND AN, B., 2020. Combating noisy labels by agreement: A joint training method with co-regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 13726–13735. (cited on pages 105 and 106)

WELINDER, P.; BRANSON, S.; MITA, T.; WAH, C.; SCHROFF, F.; BELONGIE, S.; AND PERONA, P., 2010. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology. (cited on pages xv, 4, 5, 35, and 36)

WELLING, M., 2009. Herding dynamic weights for partially observed random field models. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, 599–606. (cited on page 19)

WU, L.; LIU, B.; STONE, P.; AND LIU, Q., 2020. Firefly neural architecture descent: a general approach for growing neural networks. In *Advances in Neural Information Processing Systems*, 22373–22383. Curran Associates, Inc. (cited on page 7)

WU, Y.; CHEN, Y.; WANG, L.; YE, Y.; LIU, Z.; GUO, Y.; AND FU, Y., 2019. Large scale incremental learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 374–382. (cited on pages 90 and 91)

XIA ET AL., X., 2019. Are anchor points really indispensable in label-noise learning? In *Advances in Neural Information Processing Systems*, 6835–6846. (cited on pages 21, 96, 105, and 106)

XIAO, T.; XIA, T.; YANG, Y.; HUANG, C.; AND WANG, X., 2015. Learning from massive noisy labeled data for image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2691–2699. (cited on pages 104 and 106)

XU, Y.; XIE, L.; ZHANG, X.; CHEN, X.; QI, G.-J.; TIAN, Q.; AND XIONG, H., 2020. Pc-darts: Partial channel connections for memory-efficient architecture search. In *International Conference on Learning Representations*. (cited on pages 21, 98, and 106)

YANG, H.; TIANYI ZHOU, J.; ZHANG, Y.; GAO, B.-B.; WU, J.; AND CAI, J., 2016. Exploit bounding box annotations for multi-label object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 280–288. (cited on page 55)

YE, H.-J.; HU, H.; ZHAN, D.-C.; AND SHA, F., 2018. Learning embedding adaptation for few-shot learning. *arXiv preprint arXiv:1812.03664*, (2018). (cited on pages 15, 34, and 36)

YI, K. AND WU, J., 2019. Probabilistic end-to-end noise correction for learning with noisy labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7017–7025. (cited on pages 21 and 105)

YU, L.; TWARDOWSKI, B.; LIU, X.; HERRANZ, L.; WANG, K.; CHENG, Y.; JUI, S.; AND WEIJER, J. V. D., 2020. Semantic drift compensation for class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6982–6991. (cited on page 84)

YU, X.; HAN, B.; YAO, J.; NIU, G.; TSANG, I.; AND SUGIYAMA, M., 2019. How does disagreement help generalization against label corruption? In *International Conference on Machine Learning*, 7164–7173. (cited on pages 104, 106, and 107)

YUAN, L.; CHEN, D.; CHEN, Y.-L.; CODELLA, N.; DAI, X.; GAO, J.; HU, H.; HUANG, X.; LI, B.; LI, C.; LIU, C.; LIU, M.; LIU, Z.; LU, Y.; SHI, Y.; WANG, L.; WANG, J.; XIAO, B.; XIAO, Z.; YANG, J.; ZENG, M.; ZHOU, L.; AND ZHANG, P., 2021. Florence: A new foundation model for computer vision. (cited on page 4)

ZAGORUYKO, S. AND KOMODAKIS, N., 2016. Wide residual networks. In *Proceedings of the British Machine Vision Conference*, 87.1–87.12. (cited on page 70)

ZELA, A.; ELSKEN, T.; SAIKIA, T.; MARRAKCHI, Y.; BROX, T.; AND HUTTER, F., 2020a. Understanding and robustifying differentiable architecture search. In *International Conference on Learning Representations*. (cited on pages 21 and 98)

ZELA, A.; SIEMS, J.; AND HUTTER, F., 2020b. Nas-bench-1shot1: Benchmarking and dissecting one-shot neural architecture search. In *International Conference on Learning Representations*. (cited on page 106)

ZHANG, C.; CAI, Y.; LIN, G.; AND SHEN, C., 2020. Deepemd: Few-shot image classification with differentiable earth mover's distance and structured classifiers. *arXiv preprint arXiv:2003.06777*, (2020). (cited on pages 16, 23, 29, 34, 35, 36, 38, and 41)

ZHANG, C.; DING, H.; LIN, G.; LI, R.; WANG, C.; AND SHEN, C., 2021. Meta navigator: Search for a good adaptation policy for few-shot learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 9435–9444. (cited on page 23)

ZHANG, H. AND KONIUSZ, P., 2019. Power normalizing second-order similarity network for few-shot learning. In *Winter Conference on Applications of Computer Vision*. (cited on pages 35, 36, and 38)

ZHANG, Y.; QU, H.; CHEN, C.; AND METAXAS, D., 2019. Taming the noisy gradient: Train deep neural networks with small batch sizes. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, 4348–4354. (cited on page 63)

ZHANG, Z. AND SABUNCU, M., 2018. Generalized cross entropy loss for training deep neural networks with noisy labels. In *Advances in neural information processing systems*, 8778–8788. (cited on pages 21 and 96)

ZHAO, S.; WANG, G.; ZHANG, S.; GU, Y.; LI, Y.; SONG, Z.; XU, P.; HU, R.; CHAI, H.; AND KEUTZER, K., 2020. Multi-source distilling domain adaptation. *Proceedings of the AAAI Conference on Artificial Intelligence*, (2020), 12975–12983. (cited on page 7)

ZHOU, P.; HOU, Y.; AND FENG, J., 2018. Deep adversarial subspace clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1596–1604. (cited on page 24)

ZHU, F.; LI, H.; OUYANG, W.; YU, N.; AND WANG, X., 2017. Learning spatial regularization with image-level supervisions for multi-label image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5513–5522. (cited on page 48)

ZHU, H.; YU, J.; GUPTA, A.; SHAH, D.; HARTIKAINEN, K.; SINGH, A.; KUMAR, V.; AND LEVINE, S., 2020. The ingredients of real world robotic reinforcement learning. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rJe2syrtvS. (cited on page 2)

ZINTGRAF, L.; SHIARLI, K.; KURIN, V.; HOFMANN, K.; AND WHITESON, S., 2019. Fast context adaptation via meta-learning. In *International Conference on Machine Learning*, 7693–7702. (cited on pages 17, 63, 66, 71, and 72)

ZOPH, B. AND LE, Q., 2017. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*. (cited on pages 7, 21, and 95)

ZOPH, B.; VASUDEVAN, V.; SHLENS, J.; AND LE, Q. V., 2018. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 8697–8710. (cited on pages 7, 21, 95, 98, and 104)